

Package ‘pharmr’

May 28, 2026

Encoding UTF-8

Version 2.1.1

Date 2026-05-19

Title Interface to the 'PharmPy' 'Pharmacometrics' Library

Maintainer Rikard Nordgren <rikard.nordgren@uu.se>

Depends R (>= 3.6.0),

SystemRequirements Python (>= 3.11.0)

Imports htmltools (>= 0.5.8.1), reticulate (>= 1.38), cli, utils

Suggests testthat, magrittr, here, knitr

NeedsCompilation no

Description Interface to the 'PharmPy' 'pharmacometrics' library. The 'Reticulate' package is used to interface Python from R.

URL <https://github.com/pharmPy/pharmr>

BugReports <https://github.com/pharmPy/pharmr/issues>

License LGPL (>= 3)

Config/roxygen2/version 8.0.0

Author Rikard Nordgren [aut, cre, cph],
Stella Belin [aut, cph],
Mats O. Karlsson [sad],
Andrew C. Hooker [sad],
Xiaomei Chen [sad],
Sebastian Ueckert [sad] (ORCID:
<<https://orcid.org/0000-0002-3712-0255>>),
Simon Buatois [rev],
João A. Abrantes [rev],
Emilie Schindler [rev],
F. Hoffmann-La Roche Ltd. [fnd],
Bayer AG [fnd]

Repository CRAN

Date/Publication 2026-05-28 14:20:02 UTC

Contents

add_admid	8
add_allometry	9
add_bioavailability	10
add_cmt	11
add_covariate_effect	12
add_derivative	14
add_effect_compartment	15
add_estimation_step	16
add_iiv	18
add_indirect_effect	20
add_individual_parameter	21
add_iov	22
add_lag_time	23
add_metabolite	24
add_output_variables	24
add_parameter_uncertainty_step	25
add_pd_iiv	26
add_peripheral_compartment	27
add_pk_iiv	28
add_population_parameter	29
add_predictions	30
add_residuals	31
add_time_after_dose	32
add_time_of_last_dose	32
append_estimation_step_options	33
binarize_dataset	34
bin_observations	35
broadcast_log	36
bump_model_number	36
calculate_aic	37
calculate_bic	37
calculate_corr_from_cov	38
calculate_corr_from_prec	39
calculate_cov_from_corrse	40
calculate_cov_from_prec	41
calculate_epsilon_gradient_expression	42
calculate_eta_gradient_expression	43
calculate_eta_shrinkage	44
calculate_individual_parameter_statistics	45
calculate_individual_shrinkage	46
calculate_parameters_from_ucp	47
calculate_pk_parameters_statistics	48
calculate_prec_from_corrse	49
calculate_prec_from_cov	50
calculate_se_from_cov	51
calculate_se_from_prec	52

calculate_summary_statistic	53
calculate_ucp_scale	54
check_dataset	54
check_high_correlations	55
check_parameters_near_bounds	56
check_pharmpy	57
check_setup	57
cholesky_decompose	57
cleanup_model	58
convert_model	59
convert_unit	60
create_basic_kpd_model	60
create_basic_pd_model	61
create_basic_pk_model	62
create_config_template	63
create_joint_distribution	63
create_report	64
create_rng	64
create_symbol	65
deidentify_data	66
display_odes	66
drop_columns	67
drop_dropped_columns	68
evaluate_epsilon_gradient	68
evaluate_eta_gradient	69
evaluate_expression	70
evaluate_individual_prediction	71
evaluate_population_prediction	72
evaluate_weighted_residuals	73
expand_additional_doses	74
export_model_files	74
filter_dataset	75
find_clearance_parameters	76
find_volume_parameters	76
fit	77
fix_or_unfix_parameters	78
fix_parameters	79
fix_parameters_to	80
get_admid	81
get_baselines	81
get_bioavailability	82
get_central_volume_and_clearance	82
get_cmt	83
get_column_name	83
get_concentration_parameters_from_data	84
get_config_path	85
get_covariate_baselines	85
get_covariate_effects	86

get_doseid	86
get_doses	87
get_dv_symbol	88
get_evid	88
get_ids	89
get_individual_parameters	89
get_individual_prediction_expression	90
get_initial_conditions	91
get_lag_times	92
get_mdv	92
get_model_code	93
get_model_covariates	93
get_mu_connected_to_parameter	94
get_nested_model	95
get_number_of_individuals	96
get_number_of_observations	97
get_number_of_observations_per_individual	98
get_number_of_peripheral_compartments	99
get_number_of_transit_compartments	99
get_observations	100
get_observation_expression	101
get_omegas	101
get_parameter_rv	102
get_pd_parameters	103
get_pk_parameters	104
get_population_prediction_expression	105
get_rv_parameters	105
get_sigmas	106
get_thetas	107
get_unit_of	108
get_zero_order_inputs	108
greekify_model	109
has_additive_error_model	110
has_combined_error_model	111
has_covariate_effect	112
has_first_order_absorption	112
has_first_order_elimination	113
has_instantaneous_absorption	114
has_linear_odes	114
has_linear_odes_with_real_eigenvalues	115
has_michaelis_menten_elimination	116
has_mixed_mm_fo_elimination	116
has_mu_reference	117
has_odes	118
has_presystemic_metabolite	118
has_proportional_error_model	119
has_random_effect	120
has_seq_zo_fo_absorption	121

has_weibull_absorption	121
has_weighted_error_model	122
has_zero_order_absorption	122
has_zero_order_elimination	123
infer_datatypes	124
insert_ebes_into_dataset	124
install_pharmpy	125
install_pharmpy_devel	126
is_binary	126
is_linearized	127
is_real	128
is_simulation_model	128
is_strictness_fulfilled	129
list_models	130
list_time_varying_covariates	130
load_dataset	131
load_example_model	132
load_example_modelfit_results	132
make_declarative	133
map_eta_parameters	134
mu_reference_model	134
omit_data	135
open_context	136
plot_abs_cwres_vs_ipred	136
plot_cwres_vs_idv	137
plot_dv_vs_ipred	138
plot_dv_vs_pred	139
plot_eta_distributions	139
plot_individual_predictions	140
plot_iofv_vs_iofv	141
plot_transformed_eta_distributions	142
plot_vpc	142
predict_influential_individuals	144
predict_influential_outliers	144
predict_outliers	145
print_fit_summary	146
print_log	147
print_model_code	147
print_model_symbols	148
print_pharmpy_version	148
read_dataset_from_datainfo	149
read_model	149
read_modelfit_results	150
read_model_from_string	150
read_results	151
remove_bioavailability	152
remove_covariate_effect	153
remove_derivative	153

remove_error_model	154
remove_estimation_step	155
remove_iiv	156
remove_iov	157
remove_lag_time	158
remove_loq_data	158
remove_parameter_uncertainty_step	160
remove_peripheral_compartment	161
remove_predictions	162
remove_residuals	163
remove_unused_columns	164
remove_unused_parameters_and_rvs	164
rename_symbols	165
replace_fixed_thetas	165
replace_non_random_rvs	166
resample_data	166
reset_index	167
reset_indices_results	168
retrieve_model	168
retrieve_modelfit_results	169
retrieve_models	169
run_allometry	170
run_amd	171
run_bootstrap	173
run_covsearch	174
run_estmethod	176
run_iivsearch	178
run_iovsearch	179
run_linearize	180
run_modelfit	181
run_modelrank	182
run_modelsearch	183
run_pdsearch	184
run_qa	185
run_retries	186
run_ruvsearch	187
run_simulation	188
run_structsearch	189
run_tool	190
run_vpc	191
sample_individual_estimates	192
sample_parameters_from_covariance_matrix	193
sample_parameters_uniformly	194
set_additive_error_model	195
set_baseline_effect	197
set_combined_error_model	197
set_covariates	198
set_dataset	199

set_description	200
set_direct_effect	200
set_dtbs_error_model	201
set_dvid	202
set_estimation_step	203
set_evaluation_step	204
set_first_order_absorption	205
set_first_order_elimination	206
set_iiv_on_ruv	206
set_initial_condition	207
set_initial_estimates	208
set_instantaneous_absorption	209
set_lloq_data	210
set_lower_bounds	211
set_michaelis_menten_elimination	212
set_mixed_mm_fo_elimination	213
set_name	214
set_n_transit_compartments	214
set_ode_solver	215
set_peripheral_compartments	216
set_placebo_model	217
set_power_on_ruv	218
set_property	219
set_proportional_error_model	220
set_reference_values	221
set_seq_zo_fo_absorption	222
set_simulation	223
set_time_varying_error_model	223
set_tmdd	224
set_transit_compartments	225
set_unit	226
set_upper_bounds	227
set_weibull_absorption	228
set_weighted_error_model	229
set_zero_order_absorption	229
set_zero_order_elimination	230
set_zero_order_input	231
simplify_expression	232
solve_ode_system	232
split_joint_distribution	233
summarize_modelfit_results	234
transform_biq	234
transform_etas_boxcox	236
transform_etas_john_draper	237
transform_etas_tdist	238
translate_nmtran_time	239
unconstrain_parameters	239
undrop_columns	240

unfix_parameters	241
unfix_parameters_to	242
unload_dataset	243
update_initial_individual_estimates	243
use_thetas_for_error_stdev	244
write_csv	245
write_dataset	245
write_model	246
write_results	247

Index **248**

add_admid	<i>add_admid</i>
-----------	------------------

Description

Add an admid column to the model dataset and datainfo. Dependent on the presence of a CMT column in order to add admid correctly.

When generated, admids of events in between doses is set to the last used admid.

Usage

```
add_admid(model)
```

Arguments

model	(Model) PharmPy model
-------	-----------------------

Value

(model : Model) Updated PharmPy model

See Also

get_admid : Get or create an admid column

get_cmt : Get or create a cmt column

add_allometry	<i>add_allometry</i>
---------------	----------------------

Description

Add allometric scaling of parameters

Add an allometric function to each listed parameter. The function will be $P=P*(X/Z)**T$ where P is the parameter, X the allometric_variable, Z the reference_value and T is a theta. Default is to automatically use clearance and volume parameters.

If there already exists a covariate effect (or allometric scaling) on a parameter with the specified allometric variable, nothing will be added.

If no allometric variable is specified, it will be extracted from the dataset based on the descriptor "body weight".

Usage

```
add_allometry(
  model,
  allometric_variable = NULL,
  reference_value = 70,
  parameters = NULL,
  initials = NULL,
  lower_bounds = NULL,
  upper_bounds = NULL,
  fixed = TRUE
)
```

Arguments

model	(Model) Pharmspy model
allometric_variable	(str or Expr (optional)) Value to use for allometry (X above)
reference_value	(numeric or str or Expr) Reference value (Z above)
parameters	(array(numeric or str or Expr) (optional)) Parameters to use or NULL (default) for all available CL, Q and V parameters
initials	(array(numeric) (optional)) Initial estimates for the exponents. Default is to use 0.75 for CL and Qs and 1 for Vs
lower_bounds	(array(numeric) (optional)) Lower bounds for the exponents. Default is 0 for all parameters
upper_bounds	(array(numeric) (optional)) Upper bounds for the exponents. Default is 2 for all parameters
fixed	(logical) Whether the exponents should be fixed

Value

(Model) Updated PharmPy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- remove_covariate_effect(model, 'CL', 'WGT')
model <- remove_covariate_effect(model, 'V', 'WGT')
model <- add_allometry(model, allometric_variable='WGT')
model$statements$before_odes

## End(Not run)
```

add_bioavailability *add_bioavailability*

Description

Add bioavailability statement for the first dose compartment of the model. Can be added as a new parameter or otherwise it will be set to 1. If added as a parameter, a logit transformation can also be applied.

Usage

```
add_bioavailability(model, add_parameter = TRUE, logit_transform = FALSE)
```

Arguments

model (Model) PharmPy model

add_parameter (logical) Add new parameter representing bioavailability or not

logit_transform (logical) Logit transform the added bioavailability parameter.

Value

(Model) Updated PharmPy model

See Also

remove_bioavailability

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- add_bioavailability(model)  
  
## End(Not run)
```

add_cmt	<i>add_cmt</i>
---------	----------------

Description

Add a CMT column to the model dataset and datainfo if not existed

In case of multiple doses, this method is dependent on the presence of an admid column to correctly number each dose.

NOTE : Existing CMT is based on datainfo type being set to 'compartment' and a column named 'CMT' can be replaced

Usage

```
add_cmt(model)
```

Arguments

model (Model) Pharmpy model

Value

(model : Model) Updated Pharmpy model

See Also

get_admid : Get or create an admid column

get_cmt : Get or create a cmt column

add_covariate_effect *add_covariate_effect*

Description

Adds covariate effect to `:class:pharmpy.model`.

The following effects have templates:

- Linear function for continuous covariates (*lin*)
- Function:

(equation could not be rendered, see API doc on website)

- Init: 0.001
- Upper:
- If median of covariate equals minimum: 100,000
- Otherwise: (equation could not be rendered, see API doc on website)
- Lower:
- If median of covariate equals maximum: -100,000
- Otherwise: (equation could not be rendered, see API doc on website)
- Linear function for categorical covariates (*cat*)
- Function:
- If covariate is the most common category:

(equation could not be rendered, see API doc on website)

- For each additional category:

(equation could not be rendered, see API doc on website)

- Init: 0.001
- Upper: 5
- Lower: -1
- (alternative) Linear function for categorical covariates (*cat2*)
- Function:
- If covariate is the most common category:

(equation could not be rendered, see API doc on website)

- For each additional category:

(equation could not be rendered, see API doc on website)

- Init: 0.001
- Upper: 6

- Lower: 0
- Piecewise linear function/"hockey-stick", continuous covariates only (*piece_lin*)
- Function:
- If $cov \leq median$:

(equation could not be rendered, see API doc on website)

- If $cov > median$:

(equation could not be rendered, see API doc on website)

- Init: 0.001
- Upper:
- For first state: (equation could not be rendered, see API doc on website)
- Otherwise: 100,000
- Lower:
- For first state: -100,000
- Otherwise: (equation could not be rendered, see API doc on website)
- Exponential function, continuous covariates only (*exp*)
- Function:

(equation could not be rendered, see API doc on website)

- Init:
- If $lower > 0.001$ or $upper < 0.001$: (equation could not be rendered, see API doc on website)
- If estimated init is 0: (equation could not be rendered, see API doc on website)
- Otherwise: 0.001
- Upper:
- If $min - median = 0$ or $max - median = 0$: 100
- Otherwise:

(equation could not be rendered, see API doc on website)

- Lower:
- If $min - median = 0$ or $max - median = 0$: 0.01
- Otherwise:

(equation could not be rendered, see API doc on website)

- Power function, continuous covariates only (*pow*)
- Function:

(equation could not be rendered, see API doc on website)

- Init: 0.001
- Upper: 100,000
- Lower: -100

Usage

```
add_covariate_effect(
  model,
  parameter,
  covariate,
  effect,
  operation = "*",
  allow_nested = FALSE
)
```

Arguments

model	(Model) Pharmpy model
parameter	(str) Name of parameter to add covariate effect to.
covariate	(str) Name of covariate.
effect	(str) Type of covariate effect. May be abbreviated covariate effect (see above) or custom.
operation	(str) Whether the covariate effect should be added or multiplied (default).
allow_nested	(logical) Whether to allow adding a covariate effect when one already exists for the input parameter-covariate pair.

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- add_covariate_effect(model, "CL", "APGR", "exp")
model$statements$before_odes$full_expression("CL")

## End(Not run)
```

add_derivative

add_derivative

Description

Add a derivative to be calculated when running the model. Currently, only derivatives with respect to the prediction is supported. Default is to add all possible ETA and EPS derivatives. First order derivatives are specified either by single string or single-element tuple. For instance `with_respect_to = "ETA_1"` or `with_respect_to = ("ETA_1",)`

Second order derivatives are specified by giving the two independent variables in a tuple of tuples. For instance `with_respect_to ((ETA_1, EPS_1),)`

Multiple derivatives can be specified within a tuple. For instance ((ETA_1, EPS_1), "ETA_1")
Currently, only ETAs and EPSILONS are supported

Usage

```
add_derivative(model, with_respect_to = NULL)
```

Arguments

`model` (Model) PharmPy model
`with_respect_to` (array(array(str) or str) or str (optional)) Parameter name(s) to use as independent variables. Default is NULL.

Value

(Model) Updated PharmPy model

add_effect_compartment
add_effect_compartment

Description

Add an effect compartment.

Implemented PD models are:

- Linear:

(equation could not be rendered, see API doc on website)

- Emax:

(equation could not be rendered, see API doc on website)

- Step effect:

(equation could not be rendered, see API doc on website)

- Sigmoidal:

(equation could not be rendered, see API doc on website)

- Log-linear:

(equation could not be rendered, see API doc on website)

(equation could not be rendered, see API doc on website)

Usage

```
add_effect_compartment(model, expr)
```

Arguments

model (Model) PharmPy model
expr (str) Name of the PD effect function.

Value

(Model) Updated PharmPy model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- add_effect_compartment(model, "linear")  
model$statements$ode_system$find_compartment("EFFECT")  
  
## End(Not run)
```

add_estimation_step *add_estimation_step*

Description

Add estimation step

Adds estimation step for a model in a given index. Methods currently supported are: FO, FOCE, ITS, LAPLACE, IMPMAP, IMP, SAEM

Usage

```
add_estimation_step(  
  model,  
  method,  
  idx = NULL,  
  interaction = FALSE,  
  parameter_uncertainty_method = NULL,  
  evaluation = FALSE,  
  maximum_evaluations = NULL,  
  laplace = FALSE,  
  isample = NULL,  
  niter = NULL,  
  auto = NULL,  
  keep_every_nth_iter = NULL,  
  residuals = c(),  
  predictions = c(),  
  solver = NULL,  
  solver_rtol = NULL,  
  solver_atol = NULL,
```

```

    tool_options = {
  },
  derivatives = c(),
  individual_eta_samples = FALSE
)

```

Arguments

model	(Model) Pharmpy model
method	(str) estimation method to change to
idx	(numeric (optional)) index of estimation step (starting from 0), default is NULL (adds step at the end)
interaction	(logical) See <code>:class:~pharmpy.model.EstimationStep</code> for more information on options
parameter_uncertainty_method	(str (optional)) See above
evaluation	(logical) See above
maximum_evaluations	(numeric (optional)) See above
laplace	(logical) See above
isample	(numeric (optional)) See above
niter	(numeric (optional)) See above
auto	(logical (optional)) See above
keep_every_nth_iter	(numeric (optional)) See above
residuals	(array(str)) See above
predictions	(array(str)) See above
solver	(str (optional)) See above
solver_rtol	(numeric (optional)) See above
solver_atol	(numeric (optional)) See above
tool_options	(list(str=any)) See above
derivatives	(array(array(Expr))) See above
individual_eta_samples	(logical) See above

Value

(Model) Updated Pharmpy model

See Also

```

set_estimation_step
remove_estimation_step
append_estimation_step_options
add_parameter_uncertainty_step
remove_parameter_uncertainty_step
set_evaluation_step

```

Examples

```

## Not run:
model <- load_example_model("pheno")
opts <- list('NITER'=1000, 'ISAMPLE'=100)
model <- add_estimation_step(model, 'IMP', tool_options=opts)
ests <- model$execution_steps
length(ests)
ests[2]

## End(Not run)

```

add_iiv

add_iiv

Description

Adds IIVs to :class:pharmpy.model.

Effects that currently have templates are:

- Additive (*add*)
- Proportional (*prop*)
- Exponential (*exp*)
- Logit (*log*)
- Rescaled logit (*re_log*)

For all except exponential the operation input is not needed. Otherwise user specified input is supported. Initial estimates for new etas are 0.09.

Assuming a statement (equation could not be rendered, see API doc on website)

- Additive: (equation could not be rendered, see API doc on website)
- Proportional: (equation could not be rendered, see API doc on website)
- Exponential: (equation could not be rendered, see API doc on website)
- Logit: (equation could not be rendered, see API doc on website)
- Rescaled logit: (equation could not be rendered, see API doc on website) with (equation could not be rendered, see API doc on website)

Usage

```
add_iiv(  
  model,  
  list_of_parameters,  
  expression,  
  operation = "*",  
  initial_estimate = 0.09,  
  eta_names = NULL  
)
```

Arguments

model	(Model) PharmPy model
list_of_parameters	(array(str) or str) Name/names of parameter to add new IIVs to.
expression	(array(str) or str) Effect/effects on eta. Either abbreviated (see above) or custom.
operation	(str) Whether the new IIV should be added or multiplied (default).
initial_estimate	(numeric) Value of initial estimate of parameter. Default is 0.09
eta_names	(array(str) (optional)) Custom name/names of new eta

Value

(Model) Updated PharmPy model

See Also

```
add_pk_iiv  
add_iov  
remove_iiv  
remove_iov
```

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- remove_iiv(model, "CL")  
model <- add_iiv(model, "CL", "add")  
model$statements$find_assignment("CL")  
  
## End(Not run)
```

add_indirect_effect *add_indirect_effect*

Description

Add indirect (turnover) effect

The concentration (equation could not be rendered, see API doc on website)

- Production:

(equation could not be rendered, see API doc on website)

- Degradation:

(equation could not be rendered, see API doc on website)

(equation could not be rendered, see API doc on website) Baseline (equation could not be rendered, see API doc on website)

Models:

- Linear:

(equation could not be rendered, see API doc on website)

- Emax:

(equation could not be rendered, see API doc on website)

- Sigmoidal:

(equation could not be rendered, see API doc on website)

Usage

```
add_indirect_effect(model, expr, prod = TRUE, variable = NULL)
```

Arguments

model	(Model) PharmPy model
expr	(str) Production (TRUE) (default) or degradation (FALSE)
prod	(logical) Name of PD effect function.
variable	(str (optional)) Name of variable to use (if NULL concentration will be used)

Value

(Model) Updated PharmPy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- add_indirect_effect(model, expr='linear', prod=TRUE)

## End(Not run)
```

```
add_individual_parameter
      add_individual_parameter
```

Description

Add an individual or pk parameter to a model

Usage

```
add_individual_parameter(model, name, init = 0.1, lower = 0)
```

Arguments

model	(Model) PharmPy model
name	(str) Name of individual/pk parameter
init	(numeric) Initial estimate of the population parameter
lower	(numeric) Lower bound for the population parameter

Value

(Model) Updated PharmPy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- add_individual_parameter(model, "KA")
model$statements$find_assignment("KA")

## End(Not run)
```

add_iov	<i>add_iov</i>
---------	----------------

Description

Adds IOVs to :class:pharmpy.model.

Initial estimate of new IOVs are 10% of the IIV eta it is based on.

Usage

```
add_iov(
    model,
    occ,
    list_of_parameters = NULL,
    eta_names = NULL,
    distribution = "disjoint"
)
```

Arguments

model	(Model) Pharmpy model
occ	(str) Name of occasion column.
list_of_parameters	(array(str) or str (optional)) List of names of parameters and random variables. Accepts random variable names, parameter names, or a mix of both.
eta_names	(array(str) or str (optional)) Custom names of new etas. Must be equal to the number of input etas times the number of categories for occasion.
distribution	(str) The distribution that should be used for the new etas. Options are 'disjoint' for disjoint normal distributions, 'joint' for joint normal distribution, 'explicit' for an explicit mix of joint and disjoint distributions, and 'same-as-iiv' for copying the distribution of IIV etas.

Value

(Model) Updated Pharmpy model

See Also

add_iiv
 add_pk_iiv
 remove_iiv
 remove_iov

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- add_iov(model, "TIME", "CL")
model$statements$find_assignment("CL")

## End(Not run)
```

add_lag_time

add_lag_time

Description

Add lag time to the dose compartment of model.

Initial estimate for lag time is set the previous lag time if available, otherwise it is set to the time of first observation/2.

Usage

```
add_lag_time(model)
```

Arguments

model (Model) Pharmpy model

Value

(Model) Updated Pharmpy model

See Also

set_transit_compartments

remove_lag_time

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- add_lag_time(model)

## End(Not run)
```

add_metabolite *add_metabolite*

Description

Adds a metabolite compartment to a model

The flow from the central compartment to the metabolite compartment will be unidirectional.

Presystemic indicate that the metabolite compartment will be directly connected to the DEPOT. If a depot compartment is not present, one will be created.

Usage

```
add_metabolite(model, drug_dvid = 1, presystemic = FALSE)
```

Arguments

model (Model) PharmPy model
drug_dvid (numeric) DVID for drug (assuming all other DVIDs being for metabolites)
presystemic (logical) Decide whether or not to add metabolite as a presystemic fixed drug.

Value

(Model) Updated PharmPy model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- add_metabolite(model)  
  
## End(Not run)
```

add_output_variables *add_output_variables*

Description

Add output variables to an execution step

Usage

```
add_output_variables(model, variables, append = TRUE)
```

Arguments

model (Model) Pharmpy model
variables (array(str)) List of variables to add
append (logical) Set to false to overwrite all variables

Value

(Model) Updated Pharmpy model

See Also

add_predictions
add_residuals
remove_predictions
remove_residuals

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model$execution_steps[-1].variables  
model <- add_output_variables(model, c('CL'))  
model$execution_steps[-1].variables  
  
## End(Not run)
```

add_parameter_uncertainty_step
add_parameter_uncertainty_step

Description

Adds parameter uncertainty step to the final estimation step

Usage

```
add_parameter_uncertainty_step(model, parameter_uncertainty_method)
```

Arguments

model (Model) Pharmpy model
parameter_uncertainty_method
(str) Parameter uncertainty method to use

Value

(Model) Update Pharmpy model

See Also

```
add_estimation_step
set_estimation_step
remove_estimation_step
append_estimation_step_options
remove_parameter_uncertainty_step
set_evaluation_step
```

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_estimation_step(model, 'FOCE', parameter_uncertainty_method=NULL)
model <- add_parameter_uncertainty_step(model, 'SANDWICH')
ests <- model$execution_steps
ests[1]

## End(Not run)
```

add_pd_iiv

add_pd_iiv

Description

Adds IIVs to all PD parameters in :class:pharmpy.model.

Usage

```
add_pd_iiv(model, initial_estimate = 0.09)
```

Arguments

```
model          (Model) Pharmpy model
initial_estimate (numeric) Value of initial estimate of parameter. Default is 0.09
```

Value

(Model) Updated Pharmpy model

See Also

```
add_iiv
add_iov
remove_iiv
remove_iov
```

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_direct_effect(model, 'emax')
model$statements$find_assignment("EC_50")
model <- add_pd_iiv(model)
model$statements$find_assignment("EC_50")

## End(Not run)
```

```
add_peripheral_compartment
      add_peripheral_compartment
```

Description

Add a peripheral distribution compartment to model

The rate of flow from the central to the peripheral compartment will be parameterized as QP_n / VC where VC is the volume of the central compartment. The rate of flow from the peripheral to the central compartment will be parameterized as QP_n / VP_n where VP_n is the volume of the added peripheral compartment.

If name is set, the peripheral compartment will be added to the compartment with the specified name instead.

Initial estimates:

```
===== n =====
1 (equation could not be rendered, see API doc on website) 2 (equation could not be rendered, see
API doc on website) == =====
```

Usage

```
add_peripheral_compartment(model, name = NULL)
```

Arguments

model	(Model) Pharmpy model
name	(str (optional)) Name of compartment to add peripheral to.

Value

(Model) Updated Pharmpy model

See Also

```
set_peripheral_compartment
remove_peripheral_compartment
```

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- add_peripheral_compartment(model)
model$statements$ode_system

## End(Not run)
```

add_pk_iiv

add_pk_iiv

Description

Adds IIVs to all PK parameters in `:class:pharmpy.model`.

Will add exponential IIVs to all parameters that are included in the ODE.

Usage

```
add_pk_iiv(model, initial_estimate = 0.09)
```

Arguments

`model` (Model) Pharmpy model
`initial_estimate` (numeric) Value of initial estimate of parameter. Default is 0.09

Value

(Model) Updated Pharmpy model

See Also

`add_iiv`
`add_iov`
`remove_iiv`
`remove_iov`

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_first_order_absorption(model)
model$statements$find_assignment("MAT")
model <- add_pk_iiv(model)
model$statements$find_assignment("MAT")

## End(Not run)
```

```
add_population_parameter  
    add_population_parameter
```

Description

Add a new population parameter to the model

Usage

```
add_population_parameter(  
  model,  
  name,  
  init,  
  lower = NULL,  
  upper = NULL,  
  fix = FALSE  
)
```

Arguments

model	(Model) Pharmpy model
name	(str) Name of the new parameter
init	(numeric) Initial estimate of the new parameter
lower	(numeric (optional)) Lower bound of the new parameter
upper	(numeric (optional)) Upper bound of the new parameter
fix	(logical) Should the new parameter be fixed?

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- add_population_parameter(model, 'POP_KA', 2)  
model$parameters  
  
## End(Not run)
```

add_predictions *add_predictions*

Description

Add predictions and/or residuals
Add predictions to estimation step.

Usage

```
add_predictions(model, pred)
```

Arguments

model (Model) Pharmpy model
pred (array(str)) List of predictions (e.g. c('IPRED', 'PRED'))

Value

(Model) Updated Pharmpy model

See Also

remove_predictions
remove_residuals
set_estimation_step
add_estimation_step
remove_estimation_step
append_estimation_step_options
add_parameter_uncertainty_step
remove_parameter_uncertainty_step

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model$execution_steps[-1].predictions  
model <- add_predictions(model, c('IPRED'))  
model$execution_steps[-1].predictions  
  
## End(Not run)
```

add_residuals	<i>add_residuals</i>
---------------	----------------------

Description

Add predictions and/or residuals

Add residuals to estimation step.

Added residual variable(s) need to be one of the following : c('RES', 'IRES', 'WRES', 'IWRES', 'CWRES')

Usage

```
add_residuals(model, res)
```

Arguments

model	(Model) PharmPy model
res	(array(str)) List of residuals (e.g. c('CWRES'))

Value

(Model) Updated PharmPy model

See Also

remove_predictions
remove_residuals
set_estimation_step
add_estimation_step
remove_estimation_step
append_estimation_step_options
add_parameter_uncertainty_step
remove_parameter_uncertainty_step

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model$execution_steps[-1].residuals  
model <- add_residuals(model, c('WRES'))  
model$execution_steps[-1].residuals  
  
## End(Not run)
```

add_time_after_dose *add_time_after_dose*

Description

Calculate and add a TAD column to the dataset

Usage

```
add_time_after_dose(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) Updated PharmPy model

See Also

add_time_of_last_dose : Add time of last dose to model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- add_time_after_dose(model)  
  
## End(Not run)
```

add_time_of_last_dose *add_time_of_last_dose*

Description

Add a variable for time of last dose to the model

Usage

```
add_time_of_last_dose(model, name = "TDOSE")
```

Arguments

model (Model) PharmPy model
name (str) Name of time of last dose variable

Value

(Model) Updated Pharmpy model

See Also

add_time_after_dose : Add time after dose to dataset

append_estimation_step_options
append_estimation_step_options

Description

Append estimation step options
Appends options to an existing estimation step.

Usage

```
append_estimation_step_options(model, tool_options, idx = -1)
```

Arguments

model	(Model) Pharmpy model
tool_options	(list(str=any)) any additional tool specific options
idx	(numeric) index of estimation step (starting from 0). Default is the last step

Value

(Model) Updated Pharmpy model

See Also

add_estimation_step
set_estimation_step
remove_estimation_step
add_parameter_uncertainty_step
remove_parameter_uncertainty_step
set_evaluation_step

Examples

```
## Not run:
model <- load_example_model("pheno")
opts <- list('NITER'=1000, 'ISAMPLE'=100)
model <- append_estimation_step_options(model, tool_options=opts, idx=0)
est <- model$execution_steps[1]
length(est$tool_options)

## End(Not run)
```

<code>binarize_dataset</code>	<i>binarize_dataset</i>
-------------------------------	-------------------------

Description

Binarize dataset

Will create one column per category if specified, otherwise for all columns except for the last one.

Will also update datainfo so that new columns have type covariate and is categorical.

Usage

```
binarize_dataset(model, columns, keep = FALSE, all_levels = FALSE)
```

Arguments

<code>model</code>	(Model) Pharmpy model
<code>columns</code>	(array(str) (optional)) The columns to binarize or NULL for all marked as categorical
<code>keep</code>	(logical) Keep the original column in dataset (default is FALSE)
<code>all_levels</code>	(logical) Create one column per level, otherwise skip last value (default is FALSE)

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- binarize_dataset(model, c('APGR'))
model$dataset

## End(Not run)
```

```
bin_observations      bin_observations
```

Description

Bin all observations on the independent variable

Available binning methods:

Method	Description
equal_width	Bins with equal width based on the idv
equal_number	Bins containing an equal number of observations

Usage

```
bin_observations(model, method, nbins)
```

Arguments

model	(Model) Pharmpy model
method	(str) Name of the binning method to use
nbins	(numeric) The number of bins wanted

Value

(data.frame) A series of bin ids indexed on the original record index of the dataset vector A vector of bin edges

Examples

```
## Not run:
model <- load_example_model("pheno")
bins, boundaries <- bin_observations(model, method="equal_width", nbins=10)
bins
boundaries

## End(Not run)
```

<code>broadcast_log</code>	<i>broadcast_log</i>
----------------------------	----------------------

Description

Broadcast the log of a context

Default is to use the same broadcaster, but optionally another broadcaster could be used.

Usage

```
broadcast_log(context, broadcaster = NULL)
```

Arguments

<code>context</code>	(Context) Broadcast the log of this context
<code>broadcaster</code>	(str (optional)) Name of the broadcaster to use. Default is to use the same as was original used.

<code>bump_model_number</code>	<i>bump_model_number</i>
--------------------------------	--------------------------

Description

If the model name ends in a number increase it

If path is set increase the number until no file exists with the same name in path. If model name does not end in a number do nothing.

Usage

```
bump_model_number(model, path = NULL)
```

Arguments

<code>model</code>	(Model) Pharmpy model
<code>path</code>	(str (optional)) Default is to not look for files.

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- model$replace(name="run2")
model <- bump_model_number(model)
model$name

## End(Not run)
```

calculate_aic	<i>calculate_aic</i>
---------------	----------------------

Description

Calculate AIC

$AIC = -2LL + 2 * n_{estimated_parameters}$

Usage

```
calculate_aic(model, likelihood)
```

Arguments

model		(Model) PharmPy model
likelihood		(numeric) -2LL

Value

(numeric) AIC of model fit

calculate_bic	<i>calculate_bic</i>
---------------	----------------------

Description

Calculate BIC

Different variations of the BIC can be calculated:

- | mixed (default) | $BIC = -2LL + n_{random_parameters} * \log(n_{individuals}) + | n_{fixed_parameters} * \log(n_{observations})$
- | fixed | $BIC = -2LL + n_{estimated_parameters} * \log(n_{observations})$
- | random | $BIC = -2LL + n_{estimated_parameters} * \log(n_{individuals})$
- | iiv | $BIC = -2LL + n_{estimated_iiv_omega_parameters} * \log(n_{individuals})$

Usage

```
calculate_bic(model, likelihood, type = "mixed")
```

Arguments

model	(Model) Pharmpy model
likelihood	(numeric) -2LL to use
type	(str) Type of BIC to calculate. Default is the mixed effects.

Value

(numeric) BIC of model fit

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
ofv <- results$ofv
calculate_bic(model, ofv)
calculate_bic(model, ofv, type='fixed')
calculate_bic(model, ofv, type='random')
calculate_bic(model, ofv, type='iiv')

## End(Not run)
```

```
calculate_corr_from_cov
      calculate_corr_from_cov
```

Description

Calculate correlation matrix from a covariance matrix

Usage

```
calculate_corr_from_cov(cov)
```

Arguments

cov	(data.frame) Covariance matrix
-----	--------------------------------

Value

(data.frame) Correlation matrix

See Also

`calculate_se_from_cov` : Standard errors from covariance matrix
`calculate_se_from_prec` : Standard errors from precision matrix
`calculate_cov_from_prec` : Covariance matrix from precision matrix
`calculate_cov_from_corrse` : Covariance matrix from correlation matrix and standard errors
`calculate_prec_from_cov` : Precision matrix from covariance matrix
`calculate_prec_from_corrse` : Precision matrix from correlation matrix and standard errors
`calculate_corr_from_prec` : Correlation matrix from precision matrix

Examples

```
## Not run:  
results <- load_example_modelfit_results("pheno")  
cov <- results$covariance_matrix  
cov  
calculate_corr_from_cov(cov)  
  
## End(Not run)
```

`calculate_corr_from_prec`
calculate_corr_from_prec

Description

Calculate correlation matrix from a precision matrix

Usage

```
calculate_corr_from_prec(precision_matrix)
```

Arguments

`precision_matrix`
(data.frame) Precision matrix

Value

(data.frame) Correlation matrix

See Also

`calculate_se_from_cov` : Standard errors from covariance matrix
`calculate_se_from_prec` : Standard errors from precision matrix
`calculate_corr_from_cov` : Correlation matrix from covariance matrix
`calculate_cov_from_prec` : Covariance matrix from precision matrix
`calculate_cov_from_corrse` : Covariance matrix from correlation matrix and standard errors
`calculate_prec_from_cov` : Precision matrix from covariance matrix
`calculate_prec_from_corrse` : Precision matrix from correlation matrix and standard errors

Examples

```
## Not run:  
results <- load_example_modelfit_results("pheno")  
prec <- results$precision_matrix  
prec  
calculate_corr_from_prec(prec)  
  
## End(Not run)
```

```
calculate_cov_from_corrse  
      calculate_cov_from_corrse
```

Description

Calculate covariance matrix from a correlation matrix and standard errors

Usage

```
calculate_cov_from_corrse(corr, se)
```

Arguments

`corr` (data.frame) Correlation matrix
`se` (array) Standard errors

Value

(data.frame) Covariance matrix

See Also

`calculate_se_from_cov` : Standard errors from covariance matrix
`calculate_se_from_prec` : Standard errors from precision matrix
`calculate_corr_from_cov` : Correlation matrix from covariance matrix
`calculate_cov_from_prec` : Covariance matrix from precision matrix
`calculate_prec_from_cov` : Precision matrix from covariance matrix
`calculate_prec_from_corrse` : Precision matrix from correlation matrix and standard errors
`calculate_corr_from_prec` : Correlation matrix from precision matrix

Examples

```
## Not run:  
results <- load_example_modelfit_results("pheno")  
corr <- results$correlation_matrix  
se <- results$standard_errors  
corr  
calculate_cov_from_corrse(corr, se)  
  
## End(Not run)
```

```
calculate_cov_from_prec  
      calculate_cov_from_prec
```

Description

Calculate covariance matrix from a precision matrix

Usage

```
calculate_cov_from_prec(precision_matrix)
```

Arguments

```
precision_matrix  
      (data.frame) Precision matrix
```

Value

(data.frame) Covariance matrix

See Also

calculate_se_from_cov : Standard errors from covariance matrix
calculate_se_from_prec : Standard errors from precision matrix
calculate_corr_from_cov : Correlation matrix from covariance matrix
calculate_cov_from_corrse : Covariance matrix from correlation matrix and standard errors
calculate_prec_from_cov : Precision matrix from covariance matrix
calculate_prec_from_corrse : Precision matrix from correlation matrix and standard errors
calculate_corr_from_prec : Correlation matrix from precision matrix

Examples

```
## Not run:  
results <- load_example_modelfit_results("pheno")  
prec <- results$precision_matrix  
prec  
calculate_cov_from_prec(prec)  
  
## End(Not run)
```

```
calculate_epsilon_gradient_expression  
    calculate_epsilon_gradient_expression
```

Description

Calculate the symbolic expression for the epsilon gradient
This function currently only support models without ODE systems

Usage

```
calculate_epsilon_gradient_expression(model)
```

Arguments

model (Model) PharmPy model

Value

(Expression) Symbolic expression

See Also

calculate_eta_gradient_expression : Eta gradient

Examples

```
## Not run:  
model <- load_example_model("pheno_linear")  
calculate_epsilon_gradient_expression(model)  
  
## End(Not run)
```

```
calculate_eta_gradient_expression  
    calculate_eta_gradient_expression
```

Description

Calculate the symbolic expression for the eta gradient

This function currently only support models without ODE systems

Usage

```
calculate_eta_gradient_expression(model)
```

Arguments

model (Model) PharmPy model

Value

(Expression) Symbolic expression

See Also

calculate_epsilon_gradient_expression : Epsilon gradient

Examples

```
## Not run:  
model <- load_example_model("pheno_linear")  
calculate_eta_gradient_expression(model)  
  
## End(Not run)
```

```
calculate_eta_shrinkage  
    calculate_eta_shrinkage
```

Description

Calculate eta shrinkage for each eta

Usage

```
calculate_eta_shrinkage(  
  model,  
  parameter_estimates,  
  individual_estimates,  
  sd = FALSE  
)
```

Arguments

model	(Model) PharmPy model
parameter_estimates	(array) Parameter estimates
individual_estimates	(data.frame) Table of individual (eta) estimates
sd	(logical) Calculate shrinkage on the standard deviation scale (default is to calculate on the variance scale)

Value

(Series) Shrinkage for each eta

See Also

calculate_individual_shrinkage

Examples

```
## Not run:  
model <- load_example_model("pheno")  
results <- load_example_model_fit_results("pheno")  
pe <- results$parameter_estimates  
ie <- results$individual_estimates  
calculate_eta_shrinkage(model, pe, ie)  
calculate_eta_shrinkage(model, pe, ie, sd=TRUE)  
  
## End(Not run)
```

```
calculate_individual_parameter_statistics  
    calculate_individual_parameter_statistics
```

Description

Calculate statistics for individual parameters

Calculate the mean (expected value of the distribution), variance (variance of the distribution) and standard error for individual parameters described by arbitrary expressions. Any dataset column or variable used in the model can be used in the expression. The exception being that variables that depends on the solution of the ODE system cannot be used. If covariates are used in the expression the statistics of the parameter is calculated at the median value of each covariate as well as at the 5:th and 95:th percentiles. If no parameter uncertainty is available for the model the standard error will not be calculated.

Usage

```
calculate_individual_parameter_statistics(  
    model,  
    expr_or_exprs,  
    parameter_estimates,  
    covariance_matrix = NULL,  
    seed = 1234  
)
```

Arguments

model	(Model) PharmPy model
expr_or_exprs	(array(BooleanExpr) or array(Expr) or array(str) or BooleanExpr or Expr or str) expression or iterable of str or expressions Expressions or equations for parameters of interest. If equations are used the names of the left hand sides will be used as the names of the parameters.
parameter_estimates	(list(str=numeric)) Parameter estimates
covariance_matrix	(data.frame (optional)) Parameter uncertainty covariance matrix
seed	(numeric) Random number generator or integer seed

Value

(data.frame) A DataFrame of statistics indexed on parameter and covariate value.

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
rng <- create_rng(23)
pe <- results$parameter_estimates
cov <- results$covariance_matrix
calculate_individual_parameter_statistics(model, "K=CL/V", pe, cov, seed=rng)

## End(Not run)
```

```
calculate_individual_shrinkage
      calculate_individual_shrinkage
```

Description

Calculate the individual eta-shrinkage

Definition: $\text{ieta_shr} = (\text{var}(\text{eta}) / \text{omega})$

Usage

```
calculate_individual_shrinkage(
  model,
  parameter_estimates,
  individual_estimates_covariance
)
```

Arguments

`model` (Model) PharmPy model

`parameter_estimates` (array) Parameter estimates of model

`individual_estimates_covariance` (data.frame) Uncertainty covariance matrices of individual estimates

Value

(DataFrame) Shrinkage for each eta and individual

See Also

`calculate_eta_shrinkage`

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_model_fit_results("pheno")
pe <- results$parameter_estimates
covs <- results$individual_estimates_covariance
calculate_individual_shrinkage(model, pe, covs)

## End(Not run)
```

```
calculate_parameters_from_ucp
      calculate_parameters_from_ucp
```

Description

Scale parameter values from ucp to normal scale

Usage

```
calculate_parameters_from_ucp(model, scale, ucps)
```

Arguments

model	(Model) PharmPy model
scale	(UCPScale) A parameter scale
ucps	(array or list(str=numeric)) Series of parameter values

Value

(data.frame) Parameters on the normal scale

See Also

calculate_ucp_scale : Calculate the scale for conversion from ucps

Examples

```
## Not run:
model <- load_example_model("pheno")
scale <- calculate_ucp_scale(model)
values <- list('POP_CL'=0.1, 'POP_VC'=0.1, 'COVAPGR'=0.1, 'IIV_CL'=0.1, 'IIV_VC'=0.1, 'SIGMA'=0.1)
calculate_parameters_from_ucp(model, scale, values)

## End(Not run)
```

```
calculate_pk_parameters_statistics  
    calculate_pk_parameters_statistics
```

Description

Calculate statistics for common pharmacokinetic parameters

Calculate the mean (expected value of the distribution), variance (variance of the distribution) and standard error for some individual pre-defined pharmacokinetic parameters.

Usage

```
calculate_pk_parameters_statistics(  
  model,  
  parameter_estimates,  
  covariance_matrix = NULL,  
  seed = 1234  
)
```

Arguments

model	(Model) Pharmpy model
parameter_estimates	(array) Parameter estimates
covariance_matrix	(data.frame (optional)) Parameter uncertainty covariance matrix
seed	(numeric) Random number generator or seed

Value

(data.frame) A DataFrame of statistics indexed on parameter and covariate value.

See Also

calculate_individual_parameter_statistics : Calculation of statistics for arbitrary parameters

Examples

```
## Not run:  
model <- load_example_model("pheno")  
results <- load_example_modelfit_results("pheno")  
rng <- create_rng(23)  
pe <- results$parameter_estimates  
cov <- results$covariance_matrix  
calculate_pk_parameters_statistics(model, pe, cov, seed=rng)  
  
## End(Not run)
```

calculate_prec_from_corrse
calculate_prec_from_corrse

Description

Calculate precision matrix from a correlation matrix and standard errors

Usage

```
calculate_prec_from_corrse(corr, se)
```

Arguments

corr (data.frame) Correlation matrix
se (array) Standard errors

Value

(data.frame) Precision matrix

See Also

calculate_se_from_cov : Standard errors from covariance matrix
calculate_se_from_prec : Standard errors from precision matrix
calculate_corr_from_cov : Correlation matrix from covariance matrix
calculate_cov_from_prec : Covariance matrix from precision matrix
calculate_cov_from_corrse : Covariance matrix from correlation matrix and standard errors
calculate_prec_from_cov : Precision matrix from covariance matrix
calculate_corr_from_prec : Correlation matrix from precision matrix

Examples

```
## Not run:  
results <- load_example_modelfit_results("pheno")  
corr <- results$correlation_matrix  
se <- results$standard_errors  
corr  
calculate_prec_from_corrse(corr, se)  
  
## End(Not run)
```

```
calculate_prec_from_cov  
    calculate_prec_from_cov
```

Description

Calculate precision matrix from a covariance matrix

Usage

```
calculate_prec_from_cov(cov)
```

Arguments

cov (data.frame) Covariance matrix

Value

(data.frame) Precision matrix

See Also

calculate_se_from_cov : Standard errors from covariance matrix

calculate_se_from_prec : Standard errors from precision matrix

calculate_corr_from_cov : Correlation matrix from covariance matrix

calculate_cov_from_prec : Covariance matrix from precision matrix

calculate_cov_from_corrse : Covariance matrix from correlation matrix and standard errors

calculate_prec_from_corrse : Precision matrix from correlation matrix and standard errors

calculate_corr_from_prec : Correlation matrix from precision matrix

Examples

```
## Not run:  
results <- load_example_modelfit_results("pheno")  
cov <- results$covariance_matrix  
cov  
calculate_prec_from_cov(cov)  
  
## End(Not run)
```

calculate_se_from_cov *calculate_se_from_cov*

Description

Calculate standard errors from a covariance matrix

Usage

```
calculate_se_from_cov(cov)
```

Arguments

`cov` (data.frame) Input covariance matrix

Value

(data.frame) Standard errors

See Also

`calculate_se_from_prec` : Standard errors from precision matrix
`calculate_corr_from_cov` : Correlation matrix from covariance matrix
`calculate_cov_from_prec` : Covariance matrix from precision matrix
`calculate_cov_from_corrse` : Covariance matrix from correlation matrix and standard errors
`calculate_prec_from_cov` : Precision matrix from covariance matrix
`calculate_prec_from_corrse` : Precision matrix from correlation matrix and standard errors
`calculate_corr_from_prec` : Correlation matrix from precision matrix

Examples

```
## Not run:  
results <- load_example_modelfit_results("pheno")  
cov <- results$covariance_matrix  
cov  
calculate_se_from_cov(cov)  
  
## End(Not run)
```

`calculate_se_from_prec`*calculate_se_from_prec*

Description

Calculate standard errors from a precision matrix

Usage

```
calculate_se_from_prec(precision_matrix)
```

Arguments

```
precision_matrix  
  (data.frame) Input precision matrix
```

Value

(data.frame) Standard errors

See Also

`calculate_se_from_cov` : Standard errors from covariance matrix
`calculate_corr_from_cov` : Correlation matrix from covariance matrix
`calculate_cov_from_prec` : Covariance matrix from precision matrix
`calculate_cov_from_corrse` : Covariance matrix from correlation matrix and standard errors
`calculate_prec_from_cov` : Precision matrix from covariance matrix
`calculate_prec_from_corrse` : Precision matrix from correlation matrix and standard errors
`calculate_corr_from_prec` : Correlation matrix from precision matrix

Examples

```
## Not run:  
results <- load_example_modelfit_results("pheno")  
prec <- results$precision_matrix  
prec  
calculate_se_from_prec(prec)  
  
## End(Not run)
```

```
calculate_summary_statistic  
    calculate_summary_statistic
```

Description

Calculate a summary statistic for an expression over the dataset

The expression can be a dataset column name, a variable from the model, e.g. a derived covariate or a mathematical expression involving dataset columns or model variables. If a calculation involves population parameters, the initial estimates will be used.

Usage

```
calculate_summary_statistic(model, stat, expr, default = NULL)
```

Arguments

model	(Model) PharmPy model
stat	(str) The summary statistic. Can be "max", "min", "median" or "mean"
expr	(str (optional)) A mathematical expression containing column or variable names
default	(numeric (optional)) An optional default value to use in case the statistic couldn't be calculated. For example if the model has no dataset.

Value

(numeric) The summary statistic

Examples

```
## Not run:  
model <- load_example_model("pheno")  
calculate_summary_statistic(model, "max", "TIME")  
calculate_summary_statistic(model, "median", "log(WGT)")  
calculate_summary_statistic(model, "median", "TVCL")  
  
## End(Not run)
```

calculate_ucp_scale *calculate_ucp_scale*

Description

Calculate a scale for unconstrained parameters for a model

The UCPScale object can be used to calculate unconstrained parameters back into the normal parameter space.

Usage

```
calculate_ucp_scale(model)
```

Arguments

model (Model) PharmPy model

Value

(UCPScale) A scale object

See Also

calculate_parameters_from_ucp : Calculate parameters from ucp:s

Examples

```
## Not run:  
model <- load_example_model("pheno")  
scale <- calculate_ucp_scale(model)  
  
## End(Not run)
```

check_dataset *check_dataset*

Description

Check dataset for consistency across a set of rules

Usage

```
check_dataset(model, dataframe = FALSE, verbose = FALSE)
```

Arguments

model	(Model) PharmPy model
dataframe	(logical) TRUE to return a DataFrame instead of printing to the console
verbose	(logical) Print out all rules checked if TRUE else print only failed rules

Value

(data.frame) Only returns a DataFrame is dataframe=TRUE

```
check_high_correlations
      check_high_correlations
```

Description

Check for highly correlated parameter estimates

Usage

```
check_high_correlations(model, cor, limit = 0.9)
```

Arguments

model	(Model) PharmPy model
cor	(data.frame) Estimated correlation matrix
limit	(numeric) Lower limit for a high correlation

Value

(data.frame) Correlation values indexed on pairs of parameters for (absolute) correlations above limit

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_model_fit_results("pheno")
cor <- results$correlation_matrix
check_high_correlations(model, cor, limit=0.3)

## End(Not run)
```

```
check_parameters_near_bounds  
    check_parameters_near_bounds
```

Description

Check if any estimated parameter value is close to its bounds

Usage

```
check_parameters_near_bounds(  
  model,  
  values,  
  zero_limit = 0.001,  
  significant_digits = 2  
)
```

Arguments

model	(Model) PharmPy model
values	(array) Series of values with index a subset of parameter names.
zero_limit	(numeric) maximum distance to 0 bounds
significant_digits	(numeric) maximum distance to non-zero bounds in number of significant digits

Value

(data.frame) Logical Series with same index as values

Examples

```
## Not run:  
model <- load_example_model("pheno")  
results <- load_example_modelfit_results("pheno")  
check_parameters_near_bounds(model, results$parameter_estimates)  
  
## End(Not run)
```

check_pharmpy	<i>Checks version of Pharmpy/pharmr</i>
---------------	---

Description

Checks whether Pharmpy and pharmr has the same version

Usage

```
check_pharmpy(pharmpy_version)
```

Arguments

pharmpy_version
(str) version number as string

check_setup	<i>Checks setup of Pharmpy/pharmr</i>
-------------	---------------------------------------

Description

Checks if everything is setup correctly. The following things are checked:

- If Python is installed and has correct version
- If Pharmpy is available
- If Pharmpy and pharmr version matches

Usage

```
check_setup()
```

cholesky_decompose	<i>cholesky_decompose</i>
--------------------	---------------------------

Description

Cholesky decomposition of joint normally distributed random variables

Usage

```
cholesky_decompose(model, rvs = NULL)
```

Arguments

model (Model) Pharmpy model
rvs (array(str) (optional)) Names of random variables to decompose. NULL means all etas and is the default

Value

(Model) An updated Pharmpy model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- create_joint_distribution(model, c('ETA_CL', 'ETA_VC'))  
model <- cholesky_decompose(model)  
model$statements  
  
## End(Not run)
```

cleanup_model

cleanup_model

Description

Perform various cleanups of a model

This is what is currently done

- Make model statements declarative, i.e. only one assignment per symbol
- Inline all assignments of one symbol, e.g. $X = Y$
- Remove all random variables with no variability (i.e. with omegas fixed to zero)
- Put fixed thetas directly in the model statements

Usage

```
cleanup_model(model)
```

Arguments

model (Model) Pharmpy model

Value

(Model) Updated Pharmpy model

Note

When creating NONMEM code from the cleaned model PharmPy might need to add certain assignments to make it in line with what NONMEM requires.

Examples

```
## Not run:
model <- load_example_model("pheno")
model$statements
model <- cleanup_model(model)
model$statements

## End(Not run)
```

convert_model	<i>convert_model</i>
---------------	----------------------

Description

Convert model to other format

Note that the operation is not done inplace.

Usage

```
convert_model(model, to_format)
```

Arguments

model	(Model) PharmPy model
to_format	(str) Name of format to convert into. Currently supported 'generic', 'nlmixr', 'nonmem', and 'rxode'

Value

(Model) Updated PharmPy model

Examples

```
## Not run:
model <- load_example_model("pheno")
converted_model <- convert_model(model, "nlmixr")

## End(Not run)
```

convert_unit	<i>convert_unit</i>
--------------	---------------------

Description

Convert between units for a data variable

The conversion could either be handled in the model code or optionally in the dataset (if applicable).

Usage

```
convert_unit(model, variable, unit, original_unit = NULL, in_dataset = FALSE)
```

Arguments

model	(Model) Pharmpy model
variable	(str) Which variable in the dataset or the model code to convert
unit	(str or Unit) The new unit
original_unit	(str or Unit (optional)) If no original unit is available in the datainfo this will be used
in_dataset	(logical) Set to TRUE if the conversion should be done in the dataset instead of in model code

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- convert_unit(model, "WGT", "g")

## End(Not run)
```

create_basic_kpd_model	<i>create_basic_kpd_model</i>
------------------------	-------------------------------

Description

Creates a basic kpd model. The model will be a one compartment model. delay. The elimination rate will be (equation could not be rendered, see API doc on website)

Usage

```
create_basic_kpd_model(dataset_path = NULL, driver = "ir")
```

Arguments

```
dataset_path  (str (optional)) Optional path to a dataset
driver        (str) Driver variable of the KPD model. Can either be 'ir' (virtual infusion rate)
              or 'amount'.
```

Value

(Model) Pharmpy model

Examples

```
## Not run:
model <- create_basic_kpd_model()

## End(Not run)
```

create_basic_pd_model *create_basic_pd_model*

Description

Create a basic pd model.

The model will be very simple describing only a constant baseline effect with an additive error.

Usage

```
create_basic_pd_model(dataset_path = NULL)
```

Arguments

```
dataset_path  (str (optional)) Optional path to a dataset
```

Value

(Model) Pharmpy model

Examples

```
## Not run:
model <- create_basic_pd_model()

## End(Not run)
```

```
create_basic_pk_model create_basic_pk_model
```

Description

Creates a basic pk model of given type. The model will be a one compartment model, with first order elimination and in the case of oral administration first order absorption with no absorption delay. The elimination rate will be (equation could not be rendered, see API doc on website)

Usage

```
create_basic_pk_model(  
  administration = "iv",  
  dataset_path = NULL,  
  cl_init = 0.01,  
  vc_init = 1,  
  mat_init = 0.1  
)
```

Arguments

`administration` (str) Type of PK model to create. Supported are 'iv', 'oral' and 'ivoral'

`dataset_path` (str (optional)) Optional path to a dataset

`cl_init` (numeric) Initial estimate of the clearance parameter

`vc_init` (numeric) Initial estimate of the central volume parameter

`mat_init` (numeric) Initial estimate of the mean absorption time parameter (if applicable)

Value

(Model) PharmPy model

Examples

```
## Not run:  
model <- create_basic_pk_model('oral')  
  
## End(Not run)
```

```
create_config_template
    create_config_template
```

Description

Create a basic config file template

If a configuration file already exists it will not be overwritten

Usage

```
create_config_template()
```

Examples

```
## Not run:
create_config_template()

## End(Not run)
```

```
create_joint_distribution
    create_joint_distribution
```

Description

Combines some or all etas into a joint distribution.

The etas must be IIVs and cannot be fixed. Initial estimates for covariance between the etas is dependent on whether the model has results from a previous run. In that case, the correlation will be calculated from individual estimates, otherwise correlation will be set to 10%.

Usage

```
create_joint_distribution(model, rvs = NULL, individual_estimates = NULL)
```

Arguments

model	(Model) Pharnpy model
rvs	(array(str) (optional)) Sequence of etas or individual parameters. If NULL, all etas that are IIVs and non-fixed will be used (full block). NULL is default.
individual_estimates	(data.frame (optional)) Optional individual estimates to use for calculation of initial estimates

Value

(Model) Updated Pharmpy model

See Also

split_joint_distribution : split etas into separate distributions

Examples

```
## Not run:
model <- load_example_model("pheno")
model$random_variables$etas
model <- create_joint_distribution(model, c('ETA_CL', 'ETA_VC'))
model$random_variables$etas

## End(Not run)
```

create_report	<i>create_report</i>
---------------	----------------------

Description

Create standard report for results
The report will be an html created at specified path.

Usage

```
create_report(results, path)
```

Arguments

results	(Results) Results for which to create report
path	(str) Path to report file

create_rng	<i>create_rng</i>
------------	-------------------

Description

Create a new random number generator
Pharmpy functions that use random sampling take a random number generator or seed as input.
This function can be used to create a default new random number generator.

Usage

```
create_rng(seed = 1234)
```

Arguments

seed (numeric) Seed for the random number generator or NULL (default) for a randomized seed. If seed is generator it will be passed through.

Value

(Generator) Initialized numpy random number generator object

Examples

```
## Not run:
rng <- create_rng(23)
rng$standard_normal()

## End(Not run)
```

<code>create_symbol</code>	<i>create_symbol</i>
----------------------------	----------------------

Description

Create a new unique variable symbol given a model

Usage

```
create_symbol(model, stem, force_numbering = FALSE)
```

Arguments

model (Model) PharmPy model
stem (str) First part of the new variable name
force_numbering (logical) Forces addition of number to name even if variable does not exist, e.g. COVEFF → COVEFF1

Value

(Symbol) Created symbol with unique name

Examples

```
## Not run:
model <- load_example_model("pheno")
create_symbol(model, "TEMP")
create_symbol(model, "TEMP", force_numbering=TRUE)
create_symbol(model, "CL")

## End(Not run)
```

deidentify_data	<i>deidentify_data</i>
-----------------	------------------------

Description

Deidentify a dataset

Two operations are performed on the dataset:

1. All ID numbers are randomized from the range 1 to n
2. All columns containing dates will have the year changed

The year change is done by letting the earliest year in the dataset be used as a reference and by maintaining leap years. The reference year will either be 1901, 1902, 1903 or 1904 depending on its distance to the closest preceding leap year.

Usage

```
deidentify_data(df, id_column = "ID", date_columns = NULL)
```

Arguments

df	(data.frame) A dataset
id_column	(str) Name of the id column
date_columns	(array(str) (optional)) Names of all date columns

Value

(data.frame) Deidentified dataset

display_odes	<i>display_odes</i>
--------------	---------------------

Description

Displays the ordinary differential equation system

Usage

```
display_odes(model)
```

Arguments

model	(Model) Pharmpy model
-------	-----------------------

Value

(ODEDisplayer) A displayable object

Examples

```
## Not run:  
model <- load_example_model("pheno")  
display_odes(model)  
  
## End(Not run)
```

drop_columns	<i>drop_columns</i>
--------------	---------------------

Description

Drop columns from the dataset or mark as dropped

Usage

```
drop_columns(model, column_names, mark = FALSE)
```

Arguments

model	(Model) PharmPy model
column_names	(array(str) or str) List of column names or one column name to drop or mark as dropped
mark	(logical) Default is to remove column from dataset. Set this to TRUE to only mark as dropped

Value

(Model) Updated PharmPy model

See Also

drop_dropped_columns : Drop all columns marked as drop

undrop_columns : Undrop columns of model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- drop_columns(model, c('WGT', 'APGR'))  
vector(model$dataset$columns)  
  
## End(Not run)
```

drop_dropped_columns *drop_dropped_columns*

Description

Drop columns marked as dropped from the dataset

NM-TRAN date columns will not be dropped by this function even if marked as dropped. Columns not specified in the datainfo (\$INPUT for NONMEM) will also be dropped from the dataset.

Usage

```
drop_dropped_columns(model)
```

Arguments

model (Model) Pharmpy model

Value

(Model) Updated Pharmpy model

See Also

drop_columns : Drop specific columns or mark them as drop

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- drop_dropped_columns(model)
vector(model$dataset$columns)

## End(Not run)
```

evaluate_epsilon_gradient
 evaluate_epsilon_gradient

Description

Evaluate the numeric epsilon gradient

The gradient is evaluated at the current model parameter values or optionally at the given parameter values. The gradient is done for each data record in the model dataset or optionally using the dataset argument. The gradient is done at the current eta values or optionally at the given eta values.

This function currently only support models without ODE systems

Usage

```
evaluate_epsilon_gradient(
  model,
  etas = NULL,
  parameters = NULL,
  dataset = NULL
)
```

Arguments

model	(Model) PharmPy model
etas	(data.frame (optional)) Optional list of eta values
parameters	(list(str=numeric) (optional)) Optional list of parameters and values
dataset	(data.frame (optional)) Optional dataset

Value

(data.frame) Gradient

See Also

evaluate_eta_gradient : Evaluate the eta gradient

Examples

```
## Not run:
model <- load_example_model("pheno_linear")
results <- load_example_model_fit_results("pheno_linear")
etas <- results$individual_estimates
evaluate_epsilon_gradient(model, etas=etas)

## End(Not run)
```

evaluate_eta_gradient *evaluate_eta_gradient*

Description

Evaluate the numeric eta gradient

The gradient is evaluated at the current model parameter values or optionally at the given parameter values. The gradient is done for each data record in the model dataset or optionally using the dataset argument. The gradient is done at the current eta values or optionally at the given eta values.

This function currently only support models without ODE systems

Usage

```
evaluate_eta_gradient(model, etas = NULL, parameters = NULL, dataset = NULL)
```

Arguments

model	(Model) Pharmpy model
etas	(data.frame (optional)) Optional list of eta values
parameters	(list(str=numeric) (optional)) Optional list of parameters and values
dataset	(data.frame (optional)) Optional dataset

Value

(data.frame) Gradient

See Also

evaluate_epsilon_gradient : Evaluate the epsilon gradient

Examples

```
## Not run:
model <- load_example_model("pheno_linear")
results <- load_example_modelfit_results("pheno_linear")
etas <- results$individual_estimates
evaluate_eta_gradient(model, etas=etas)

## End(Not run)
```

evaluate_expression *evaluate_expression*

Description

Evaluate expression using model

Calculate the value of expression for each data record. The expression can contain dataset columns, variables in model and population parameters. If the model has parameter estimates these will be used. Initial estimates will be used for non-estimated parameters.

Usage

```
evaluate_expression(model, expression, parameter_estimates = NULL)
```

Arguments

model	(Model) Pharmpy model
expression	(str or numeric or Expr) Expression to evaluate
parameter_estimates	(list(str=numeric) (optional)) Parameter estimates to use instead of initial estimates

Value

(data.frame) A series of one evaluated value for each data record

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
pe <- results$parameter_estimates
evaluate_expression(model, "TVCL*1000", parameter_estimates=pe)

## End(Not run)
```

```
evaluate_individual_prediction
      evaluate_individual_prediction
```

Description

Evaluate the numeric individual prediction

The prediction is evaluated at the current model parameter values or optionally at the given parameter values. The evaluation is done for each data record in the model dataset or optionally using the dataset argument. The evaluation is done at the current eta values or optionally at the given eta values.

This function currently only support models without ODE systems

Usage

```
evaluate_individual_prediction(
  model,
  etas = NULL,
  parameters = NULL,
  dataset = NULL
)
```

Arguments

model	(Model) PharmPy model
etas	(data.frame (optional)) Optional list of eta values
parameters	(list(str=numeric) (optional)) Optional list of parameters and values
dataset	(data.frame (optional)) Optional dataset

Value

(data.frame) Individual predictions

See Also

evaluate_population_prediction : Evaluate the population prediction

Examples

```
## Not run:
model <- load_example_model("pheno_linear")
results <- load_example_modelfit_results("pheno_linear")
etas <- results$individual_estimates
evaluate_individual_prediction(model, etas=etas)

## End(Not run)
```

evaluate_population_prediction
evaluate_population_prediction

Description

Evaluate the numeric population prediction

The prediction is evaluated at the current model parameter values or optionally at the given parameter values. The evaluation is done for each data record in the model dataset or optionally using the dataset argument.

This function currently only support models without ODE systems

Usage

```
evaluate_population_prediction(model, parameters = NULL, dataset = NULL)
```

Arguments

model	(Model) PharmPy model
parameters	(list(str=numeric) (optional)) Optional list of parameters and values
dataset	(data.frame (optional)) Optional dataset

Value

(data.frame) Population predictions

See Also

evaluate_individual_prediction : Evaluate the individual prediction

Examples

```
## Not run:
model <- load_example_model("pheno_linear")
results <- load_example_model_fit_results("pheno_linear")
pe <- results$parameter_estimates
evaluate_population_prediction(model, parameters=list(pe))

## End(Not run)
```

```
evaluate_weighted_residuals
      evaluate_weighted_residuals
```

Description

Evaluate the weighted residuals

The residuals is evaluated at the current model parameter values or optionally at the given parameter values. The residuals is done for each data record in the model dataset or optionally using the dataset argument.

This function currently only support models without ODE systems

Usage

```
evaluate_weighted_residuals(model, parameters = NULL, dataset = NULL)
```

Arguments

model	(Model) Pharmpy model
parameters	(list(str=numeric) (optional)) Optional list of parameters and values
dataset	(data.frame (optional)) Optional dataset

Value

(data.frame) WRES

Examples

```
## Not run:
model <- load_example_model("pheno_linear")
results <- load_example_model_fit_results("pheno_linear")
parameters <- results$parameter_estimates
evaluate_weighted_residuals(model, parameters=list(parameters))

## End(Not run)
```

expand_additional_doses *expand_additional_doses*

Description

Expand additional doses into separate dose records

Usage

```
expand_additional_doses(model, flag = FALSE)
```

Arguments

model	(Model) Pharmpy model
flag	(logical) TRUE to add a boolean EXPANDED column to mark added records. In this case all columns in the original dataset will be kept. Care needs to be taken to handle the new dataset.

Value

(Model) Updated Pharmpy model

export_model_files *export_model_files*

Description

Exports all model files to specified directory.

Will export all model files generated/related to the external software used. Files will be named with model name (from context) and original suffix (e.g. model.ctl for modelsearch_run1 -> modelsearch_run1.ctl). If no suffix, file will be named model name and original name (e.g. mytab for modelsearch_run1 -> modelsearch_run1_mytab)

Usage

```
export_model_files(context, destination_path = NULL, force = FALSE)
```

Arguments

context	(Context) The context
destination_path	(str (optional)) Path to export model files to, NULL means current working directory
force	(logical) Allow file overwrite (default is FALSE)

Examples

```
## Not run:  
ctx <- open_context("myrun")  
export_model_files(ctx)  
  
## End(Not run)
```

filter_dataset	<i>filter_dataset</i>
----------------	-----------------------

Description

Filter dataset according to expr and return a model with the filtered dataset.

Example: "DVID == 1" will filter the dataset so that only the rows with DVID = 1 remain.

Usage

```
filter_dataset(model, expr)
```

Arguments

model	(Model) Pharmpy model
expr	(str) expression for dataset query

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model$dataset  
model <- filter_dataset(model, 'WGT < 1.4')  
model$dataset  
  
## End(Not run)
```

```
find_clearance_parameters  
    find_clearance_parameters
```

Description

Find clearance parameters in model

Usage

```
find_clearance_parameters(model)
```

Arguments

model (Model) PharmPy model

Value

(vector) A vector of clearance parameters

Examples

```
## Not run:  
model <- load_example_model("pheno")  
find_clearance_parameters(model)  
  
## End(Not run)
```

```
find_volume_parameters  
    find_volume_parameters
```

Description

Find volume parameters in model

Usage

```
find_volume_parameters(model)
```

Arguments

model (Model) PharmPy model

Value

(vector) A vector of volume parameters

Examples

```
## Not run:
model <- load_example_model("pheno")
find_volume_parameters(model)

## End(Not run)
```

fit

fit

Description

Fit models.

Usage

```
fit(model_or_models, esttool = NULL, name = NULL, context = NULL, ncores = 1)
```

Arguments

model_or_models	(Model or array(Model)) List of models or one single model
esttool	(str (optional)) Estimation tool to use. NULL to use default
name	(str (optional)) Name of run
context	(Context (optional)) Run in this context
ncores	(numeric) Number of cores to use for estimation

Value

(ModelfitResults | vector of ModelfitResults) ModelfitResults for the model or models

See Also

run_tool

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- fit(model)

## End(Not run)
```

fix_or_unfix_parameters
fix_or_unfix_parameters

Description

Fix or unfix parameters
Set fixedness of parameters to specified values

Usage

```
fix_or_unfix_parameters(model, parameters, strict = TRUE)
```

Arguments

model	(Model) PharmPy model
parameters	(list(str=logical)) Set fix/unfix for these parameters
strict	(logical) Whether all parameters in input need to exist in the model. Default is TRUE

Value

(Model) Updated PharmPy model

See Also

fix_parameters : Fix parameters
unfix_parameters : Unfixing parameters
fix_parameters_to : Fixing parameters and setting a new initial estimate in the same function
unfix_parameters_to : Unfixing parameters and setting a new initial estimate in the same function

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model$parameters['POP_CL']  
model <- fix_or_unfix_parameters(model, list('POP_CL'=TRUE))  
model$parameters['POP_CL']  
  
## End(Not run)
```

fix_parameters	<i>fix_parameters</i>
----------------	-----------------------

Description

Fix parameters

Fix all listed parameters

Usage

```
fix_parameters(model, parameter_names, strict = TRUE)
```

Arguments

model (Model) PharmPy model

parameter_names

(array(str) or str) one parameter name or a vector of parameter names

strict (logical) Whether all parameters in input need to exist in the model. Default is TRUE

Value

(Model) Updated PharmPy model

See Also

fix_or_unfix_parameters : Fix or unfix parameters (given boolean)

fix_parameters_to : Fixing and setting parameter initial estimates in the same function

unfix_paramaters : Unfixing parameters

unfix_paramaters_to : Unfixing parameters and setting a new initial estimate in the same function

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model$parameters['POP_CL']  
model <- fix_parameters(model, 'POP_CL')  
model$parameters['POP_CL']  
  
## End(Not run)
```

fix_parameters_to *fix_parameters_to*

Description

Fix parameters to

Fix all listed parameters to specified value/values

Usage

```
fix_parameters_to(model, inits, strict = TRUE)
```

Arguments

model (Model) Pharmpy model

inits (list(str=numeric)) Inits for all parameters to fix and set init

strict (logical) Whether all parameters in input need to exist in the model. Default is TRUE

Value

(Model) Updated Pharmpy model

See Also

fix_parameters : Fix parameters

fix_or_unfix_parameters : Fix or unfix parameters (given boolean)

unfix_paramaters : Unfixing parameters

unfix_paramaters_to : Unfixing parameters and setting a new initial estimate in the same function

Examples

```
## Not run:
model <- load_example_model("pheno")
model$parameters['POP_CL']
model <- fix_parameters_to(model, list('POP_CL'=0.5))
model$parameters['POP_CL']

## End(Not run)
```

`get_admid``get_admid`

Description

Get the admid from model dataset

If an administration column is present this will be extracted otherwise an admid column will be created based on the admids of the present doses. This is dependent on the presence of a CMT column to be generated correctly.

When generated, admids of events in between doses is set to the last used admid.

Usage

```
get_admid(model)
```

Arguments

model (Model) PharmPy model

Value

(data.frame) ADMID

`get_baselines``get_baselines`

Description

Baselines for each subject.

Baseline is taken to be the first row even if that has a missing value.

Usage

```
get_baselines(model)
```

Arguments

model (Model) PharmPy model

Value

(data.frame) Dataset with the baselines

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_baselines(model)  
  
## End(Not run)
```

`get_bioavailability` *get_bioavailability*

Description

Get bioavailability of doses for all compartments

Usage

```
get_bioavailability(model)
```

Arguments

model (Model) PharmPy model

Value

(list) Dictionary from compartment name to bioavailability expression

`get_central_volume_and_clearance`
get_central_volume_and_clearance

Description

Get the volume and clearance parameters

Usage

```
get_central_volume_and_clearance(model)
```

Arguments

model (Model) PharmPy model

Value

(sympy.Symbol) Volume symbol sympy.Symbol Clearance symbol

Examples

```
## Not run:
model <- load_example_model("pheno")
get_central_volume_and_clearance(model)

## End(Not run)
```

get_cmt	<i>get_cmt</i>
---------	----------------

Description

Get the cmt (compartment) column from the model dataset

If a cmt column is present this will be extracted otherwise a cmt column will be created. If created, multiple dose compartments are dependent on the presence of an admid type column, otherwise, dose/non-dose will be considered.

Usage

```
get_cmt(model)
```

Arguments

model (Model) PharmPy model

Value

(data.frame) CMT

get_column_name	<i>get_column_name</i>
-----------------	------------------------

Description

Retrieve the column with a certain type

If multiple columns have the same type an exception will be raised.

Usage

```
get_column_name(model, type)
```

Arguments

model (Model) PharmPy model

type (str) Column type. See :py:attr:pharmPy.model.datainfo.ColumnInfo.type

Value

(str or NULL) Name of the column. NULL if no column found

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_column_name(model, "dose")  
  
## End(Not run)
```

```
get_concentration_parameters_from_data  
    get_concentration_parameters_from_data
```

Description

Create a dataframe with concentration parameters

Note that all values are directly calculated from the dataset

Usage

```
get_concentration_parameters_from_data(model)
```

Arguments

model (Model) PharmPy model

Value

(data.frame) Concentration parameters

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_concentration_parameters_from_data(model)  
  
## End(Not run)
```

get_config_path	<i>get_config_path</i>
-----------------	------------------------

Description

Returns path to the user config path

Usage

```
get_config_path()
```

Value

(str or NULL) Path to user config or NULL if file does not exist

Examples

```
## Not run:  
get_config_path()  
  
## End(Not run)
```

get_covariate_baselines	<i>get_covariate_baselines</i>
-------------------------	--------------------------------

Description

Return a dataframe with baselines of all covariates for each id.
Baseline is taken to be the first row even if that has a missing value.

Usage

```
get_covariate_baselines(model)
```

Arguments

model (Model) PharmPy model

Value

(data.frame) covariate baselines

See Also

get_baselines : baselines for all data columns

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_covariates(model, c("WGT", "APGR"))
get_covariate_baselines(model)

## End(Not run)
```

```
get_covariate_effects  get_covariate_effects
```

Description

Return a list of all used covariates within a model

The list will have parameter name as key with a connected value as a vector of tuple(s) with (covariate, effect type, operator)

Usage

```
get_covariate_effects(model)
```

Arguments

model (Model) PharmPy model

Value

(Dictionary : Dictionary of parameters and connected covariate(s))

```
get_doseid  get_doseid
```

Description

Get a DOSEID series from the dataset with an id of each dose period starting from 1

If a a dose and observation exist at the same time point the observation will be counted towards the previous dose.

Usage

```
get_doseid(model)
```

Arguments

model (Model) PharmPy model

Value

(data.frame) DOSEIDs

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_doseid(model)  
  
## End(Not run)
```

get_doses

get_doses

Description

Get a series of all doses
Indexed with ID and TIME

Usage

```
get_doses(model)
```

Arguments

model (Model) PharmPy model

Value

(data.frame) doses

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_doses(model)  
  
## End(Not run)
```

get_dv_symbol	<i>get_dv_symbol</i>
---------------	----------------------

Description

Get the symbol for a certain dvid or dv and check that it is valid

Usage

```
get_dv_symbol(model, dv = NULL)
```

Arguments

model	(Model) PharmPy model
dv	(Expr or str or numeric (optional)) Either a dv symbol, str or dvid. If NULL (default) return the only or first dv.

Value

(Expr) DV symbol

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_dv_symbol(model, "Y")  
get_dv_symbol(model, 1)  
  
## End(Not run)
```

get_evid	<i>get_evid</i>
----------	-----------------

Description

Get the evid from model dataset

If an event column is present this will be extracted otherwise an evid column will be created.

Usage

```
get_evid(model)
```

Arguments

model	(Model) PharmPy model
-------	-----------------------

Value

(data.frame) EVID

get_ids	<i>get_ids</i>
---------	----------------

Description

Retrieve a vector of all subject ids of the dataset

Usage

```
get_ids(model)
```

Arguments

model (Model) PharmPy model

Value

(vector) All subject ids

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_ids(model)  
  
## End(Not run)
```

get_individual_parameters	<i>get_individual_parameters</i>
---------------------------	----------------------------------

Description

Retrieves all individual parameters in a :class:pharmPy.model.

By default all individual parameters will be found even ones having no random effect. The level arguments makes it possible to find only those having any random effect or only those having a certain random effect. Using the dv option will give all individual parameters affecting a certain dv. Note that the DV for PD in a PKPD model often also is affected by the PK parameters.

Usage

```
get_individual_parameters(model, level = "all", dv = NULL)
```

Arguments

<code>model</code>	(Model) PharmPy model
<code>level</code>	(str) The variability level to look for: 'iiv', 'ioi', 'random' or 'all' (default)
<code>dv</code>	(str or Expr or numeric (optional)) Name or DVID of dependent variable. NULL for all (default)

Value

(vector<str>) A vector of the parameter names as strings

See Also

`get_pd_parameters`
`get_pk_parameters`
`get_rv_parameters`
`has_random_effect`

Examples

```
## Not run:
model <- load_example_model("pheno")
get_individual_parameters(model)
get_individual_parameters(model, 'iiv')
get_individual_parameters(model, 'ioi')

## End(Not run)
```

```
get_individual_prediction_expression
      get_individual_prediction_expression
```

Description

Get the full symbolic expression for the modelled individual prediction
 This function currently only support models without ODE systems

Usage

```
get_individual_prediction_expression(model)
```

Arguments

<code>model</code>	(Model) PharmPy model
--------------------	-----------------------

Value

(Expression) Symbolic expression

See Also

`get_population_prediction_expression` : Get full symbolic expression for the population prediction

Examples

```
## Not run:  
model <- load_example_model("pheno_linear")  
get_individual_prediction_expression(model)  
  
## End(Not run)
```

`get_initial_conditions`
get_initial_conditions

Description

Get initial conditions for the ode system
Default initial conditions at t=0 for amounts is 0

Usage

```
get_initial_conditions(model, dosing = FALSE)
```

Arguments

`model` (Model) Pharnpy model
`dosing` (logical) Set to TRUE to add dosing as initial conditions

Value

(list) Initial conditions

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_initial_conditions(model)  
get_initial_conditions(model, dosing=TRUE)  
  
## End(Not run)
```

get_lag_times	<i>get_lag_times</i>
---------------	----------------------

Description

Get lag times for all compartments

Usage

```
get_lag_times(model)
```

Arguments

model (Model) PharmPy model

Value

(list) Dictionary from compartment name to lag time expression

get_mdv	<i>get_mdv</i>
---------	----------------

Description

Get MDVs from dataset

Usage

```
get_mdv(model)
```

Arguments

model (Model) PharmPy model

Value

(data.frame) MDVs

get_model_code *get_model_code*

Description

Get the model code of the underlying model language as a string

Usage

```
get_model_code(model)
```

Arguments

model (Model) PharmPy model

Value

(str) Model code

Examples

```
## Not run:  
model <- load_example_model("pheno")  
code <- get_model_code(model)  
  
## End(Not run)
```

get_model_covariates *get_model_covariates*

Description

List of covariates used in model

A covariate in the model is here defined to be a data item affecting the model prediction excluding dosing items that are not used in model code.

Usage

```
get_model_covariates(model, strings = FALSE)
```

Arguments

model (Model) PharmPy model

strings (logical) Return strings instead of symbols? FALSE (default) will give symbols

Value

(vector) Covariate symbols or names

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_model_covariates(model)  
get_model_covariates(model, strings=TRUE)  
  
## End(Not run)
```

```
get_mu_connected_to_parameter  
                                  get_mu_connected_to_parameter
```

Description

Return Mu name connected to parameter

If the given parameter is not dependent on any Mu, NULL is returned

Usage

```
get_mu_connected_to_parameter(model, parameter)
```

Arguments

model (Model) PharmPy model

parameter (str) Name of parameter which to find Mu parameter for.

Value

(str) Name of Mu parameter or NULL

get_nested_model	<i>get_nested_model</i>
------------------	-------------------------

Description

Return nested model from a pair of models

Function to get a nested model from a pair of models, NULL if neither model is nested. A model is not considered nested if:

1. They are the same model
2. They have the same number of parameters
3. The parameters of the reduced model is not a subset of the extended model
4. The dosing or DV is changed

Assumptions made:

1. Parametrization is the same
2. Parameter names are the same

Usage

```
get_nested_model(model_1, model_2)
```

Arguments

model_1	(Model) PharmPy model
model_2	(Model) PharmPy model

Value

(Model | NULL) PharmPy model or NULL

Examples

```
## Not run:  
model_1 <- load_example_model("pheno")  
model_2 <- add_peripheral_compartment(model_1)  
model_2 <- set_name(model_2, 'pheno_2')  
nested <- get_nested_model(model_1, model_2)  
nested$name  
  
## End(Not run)
```

```
get_number_of_individuals  
    get_number_of_individuals
```

Description

Retrieve the number of individuals in the model dataset

Usage

```
get_number_of_individuals(model)
```

Arguments

model (Model) PharmPy model

Value

(integer) Number of individuals in the model dataset

Note

For NONMEM models this is the number of individuals of the active dataset, i.e. after filtering of IGNORE and ACCEPT and removal of individuals with no observations.

See Also

`get_number_of_observations` : Get the number of observations in a dataset

`get_number_of_observations_per_individual` : Get the number of observations per individual in a dataset

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_number_of_individuals(model)  
  
## End(Not run)
```

```
get_number_of_observations  
    get_number_of_observations
```

Description

Retrieve the total number of observations in the model dataset

Usage

```
get_number_of_observations(model)
```

Arguments

model (Model) PharmPy model

Value

(integer) Number of observations in the model dataset

Note

For NONMEM models this is the number of observations of the active dataset, i.e. after filtering of IGNORE and ACCEPT and removal of individuals with no observations.

See Also

get_number_of_individuals : Get the number of individuals in a dataset

get_number_of_observations_per_individual : Get the number of observations per individual in a dataset

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_number_of_observations(model)  
  
## End(Not run)
```

```
get_number_of_observations_per_individual  
    get_number_of_observations_per_individual
```

Description

Number of observations for each individual

Usage

```
get_number_of_observations_per_individual(model)
```

Arguments

model (Model) PharmPy model

Value

(data.frame) Number of observations in the model dataset

Note

For NONMEM models this is the individuals and number of observations of the active dataset, i.e. after filtering of IGNORE and ACCEPT and removal of individuals with no observations.

See Also

`get_number_of_individuals` : Get the number of individuals in a dataset

`get_number_of_observations_per_individual` : Get the number of observations per individual in a dataset

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_number_of_observations_per_individual(model)  
  
## End(Not run)
```

get_number_of_peripheral_compartments
get_number_of_peripheral_compartments

Description

Return the number of peripherals compartments connected to the central compartment

Usage

`get_number_of_peripheral_compartments(model)`

Arguments

`model` (Model) PharmPy model

Value

(integer) Number of peripherals compartments

get_number_of_transit_compartments
get_number_of_transit_compartments

Description

Return the number of transit compartments in the model

Usage

`get_number_of_transit_compartments(model)`

Arguments

`model` (Model) PharmPy model

Value

(integer) Number of transit compartments

get_observations *get_observations*

Description

Get observations from dataset

Usage

```
get_observations(model, keep_index = FALSE, dv = NULL)
```

Arguments

model	(Model) Pharnpy model
keep_index	(logical) Set to TRUE if the original index should be kept. Otherwise a new index using ID and idv will be created.
dv	(Expr or str or numeric (optional)) Name or DVID of dependent variable. NULL for the default (all DVIDs)

Value

(data.frame) Observations indexed over ID and TIME

See Also

get_number_of_observations : get the number of observations
get_number_of_observations_per_individual : get the number of observations per individual

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_observations(model)  
  
## End(Not run)
```

```
get_observation_expression  
    get_observation_expression
```

Description

Get the full symbolic expression for the observation according to the model
This function currently only support models without ODE systems

Usage

```
get_observation_expression(model)
```

Arguments

model (Model) PharmPy model

Value

(Expression) Symbolic expression

Examples

```
## Not run:  
model <- load_example_model("pheno_linear")  
expr <- get_observation_expression(model)  
print(expr$unicode())  
  
## End(Not run)
```

```
get_omegas    get_omegas
```

Description

Get all omegas (variability parameters) of a model

Usage

```
get_omegas(model)
```

Arguments

model (Model) PharmPy model

Value

(Parameters) A copy of all omega parameters

See Also

get_thetas : Get theta parameters

get_sigmas : Get sigma parameters

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_omegas(model)  
  
## End(Not run)
```

get_parameter_rv *get_parameter_rv*

Description

Retrieves name of random variable in :class:pharmpy.model.Model given a parameter.

Usage

```
get_parameter_rv(model, parameter, var_type = "iiv")
```

Arguments

model	(Model) Pharmpy model
parameter	(str) Name of parameter to retrieve random variable from
var_type	(str) Variability type: iiv (default) or iov

Value

(vector<str>) A vector of random variable names for the given parameter

See Also

get_rv_parameters

has_random_effect

get_pk_parameters

get_individual_parameters

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_parameter_rv(model, 'CL')  
  
## End(Not run)
```

get_pd_parameters *get_pd_parameters*

Description

Retrieves PD parameters in `:class:pharmpy.model.Model`.

Usage

```
get_pd_parameters(model)
```

Arguments

`model` (Model) Pharmpy model

Value

(vector<str>) A vector of the PD parameter names of the given model

See Also

`get_pk_parameters`

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- set_direct_effect(model, "linear")  
get_pd_parameters(model)  
  
## End(Not run)
```

get_pk_parameters *get_pk_parameters*

Description

Retrieves PK parameters in :class:pharmpy.model.Model.

Usage

```
get_pk_parameters(model, kind = "all")
```

Arguments

model	(Model) Pharmpy model
kind	(str) The type of parameter to retrieve: 'absorption', 'distribution', 'elimination', or 'all' (default).

Value

(vector<str>) A vector of the PK parameter names of the given model

See Also

get_individual_parameters
get_rv_parameters

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_pk_parameters(model)  
get_pk_parameters(model, 'absorption')  
get_pk_parameters(model, 'distribution')  
get_pk_parameters(model, 'elimination')  
  
## End(Not run)
```

```
get_population_prediction_expression  
    get_population_prediction_expression
```

Description

Get the full symbolic expression for the modelled population prediction
This function currently only support models without ODE systems

Usage

```
get_population_prediction_expression(model)
```

Arguments

model (Model) PharmPy model

Value

(Expression) Symbolic expression

See Also

get_individual_prediction_expression : Get full symbolic expression for the individual prediction

Examples

```
## Not run:  
model <- load_example_model("pheno_linear")  
get_population_prediction_expression(model)  
  
## End(Not run)
```

```
get_rv_parameters    get_rv_parameters
```

Description

Retrieves parameters in :class:pharmPy.model.Model given a random variable.

Usage

```
get_rv_parameters(model, rv)
```

Arguments

model (Model) PharmPy model
rv (str) Name of random variable to retrieve

Value

(vector<str>) A vector of parameter names for the given random variable

See Also

has_random_effect
get_pk_parameters
get_individual_parameters

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_rv_parameters(model, 'ETA_CL')  
  
## End(Not run)
```

get_sigmas

get_sigmas

Description

Get all sigmas (residual error variability parameters) of a model

Usage

```
get_sigmas(model)
```

Arguments

model (Model) PharmPy model

Value

(Parameters) A copy of all sigma parameters

See Also

get_thetas : Get theta parameters
get_omegas : Get omega parameters

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_sigmas(model)  
  
## End(Not run)
```

get_thetas

get_thetas

Description

Get all thetas (structural parameters) of a model

Usage

```
get_thetas(model)
```

Arguments

model (Model) PharmPy model

Value

(Parameters) A copy of all theta parameters

See Also

get_omegas : Get omega parameters

get_sigmas : Get sigma parameters

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_thetas(model)  
  
## End(Not run)
```

<code>get_unit_of</code>	<i>get_unit_of</i>
--------------------------	--------------------

Description

Derive the physical unit of a variable in the model

Unit information for the dataset needs to be available. The variable can be defined in the code, a dataset column, a parameter or a random variable. Optionally units could be derived for all variables in the model.

Usage

```
get_unit_of(model, variable = NULL)
```

Arguments

<code>model</code>	(Model) Pharmpy model
<code>variable</code>	(str or Expr (optional)) Find physical unit of this variable. For NULL get a list with units for all variables defined by the model.

Value

(Unit) A unit expression

Examples

```
## Not run:
model <- load_example_model("pheno")
get_unit_of(model, "Y")
get_unit_of(model, "VC")
get_unit_of(model, "WGT")

## End(Not run)
```

<code>get_zero_order_inputs</code>	<i>get_zero_order_inputs</i>
------------------------------------	------------------------------

Description

Get zero order inputs for all compartments

Usage

```
get_zero_order_inputs(model)
```

Arguments

model (Model) Pharmpy model

Value

(Matrix) Vector of inputs

Examples

```
## Not run:  
model <- load_example_model("pheno")  
get_zero_order_inputs(model)  
  
## End(Not run)
```

greekify_model *greekify_model*

Description

Convert to using greek letters for all population parameters

Usage

```
greekify_model(model, named_subscripts = FALSE)
```

Arguments

model (Model) Pharmpy model
named_subscripts (logical) Use previous parameter names as subscripts. Default is to use integer subscripts

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model$statements  
model <- greekify_model(cleanup_model(model))  
model$statements  
  
## End(Not run)
```

```
has_additive_error_model  
    has_additive_error_model
```

Description

Check if a model has an additive error model

Multiple dependent variables are supported. By default the only (in case of one) or the first (in case of many) dependent variable is going to be checked.

Usage

```
has_additive_error_model(model, dv = NULL)
```

Arguments

model	(Model) Pharmpy model
dv	(Expr or str or numeric (optional)) Name or DVID of dependent variable. NULL for the default (first or only)

Value

(logical) TRUE if the model has an additive error model and FALSE otherwise

See Also

`has_proportional_error_model` : Check if a model has a proportional error model

`has_combined_error_model` : Check if a model has a combined error model

`has_weighted_error_model` : Check if a model has a weighted error model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
has_additive_error_model(model)  
  
## End(Not run)
```

```
has_combined_error_model  
    has_combined_error_model
```

Description

Check if a model has a combined additive and proportional error model

Multiple dependent variables are supported. By default the only (in case of one) or the first (in case of many) dependent variable is going to be checked.

Usage

```
has_combined_error_model(model, dv = NULL)
```

Arguments

model	(Model) Pharmpy model
dv	(Expr or str or numeric (optional)) Name or DVID of dependent variable. NULL for the default (first or only)

Value

(logical) TRUE if the model has a combined error model and FALSE otherwise

See Also

has_additive_error_model : Check if a model has an additive error model

has_proportional_error_model : Check if a model has a proportional error model

has_weighted_error_model : Check if a model has a weighted error model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
has_combined_error_model(model)  
  
## End(Not run)
```

has_covariate_effect *has_covariate_effect*

Description

Tests if an instance of :class:pharmpy.model has a given covariate effect.

Usage

```
has_covariate_effect(model, parameter, covariate)
```

Arguments

model	(Model) Pharmpy model
parameter	(str) Name of parameter.
covariate	(str) Name of covariate.

Value

(logical) Whether input model has a covariate effect of the input covariate on the input parameter.

Examples

```
## Not run:
model <- load_example_model("pheno")
has_covariate_effect(model, "CL", "APGR")

## End(Not run)
```

has_first_order_absorption
has_first_order_absorption

Description

Check if ode system describes a first order absorption

Currently defined as the central compartment having a unidirectional input flow from another compartment (such as depot or transit)

Usage

```
has_first_order_absorption(model)
```

Arguments

model (Model) Pharmpy model

Value

(logical) TRUE if model has first order absorption

has_first_order_elimination
has_first_order_elimination

Description

Check if the model describes first order elimination

This function relies on heuristics and will not be able to detect all possible ways of coding the first order elimination.

Usage

```
has_first_order_elimination(model)
```

Arguments

model (Model) Pharmpy model

Value

(logical) TRUE if model has describes first order elimination

Examples

```
## Not run:  
model <- load_example_model("pheno")  
has_first_order_elimination(model)  
  
## End(Not run)
```

has_instantaneous_absorption
has_instantaneous_absorption

Description

Check if ode system describes a instantaneous absorption
Defined as being a instantaneous dose directly into the central compartment

Usage

has_instantaneous_absorption(model)

Arguments

model (Model) PharmPy model

Value

(logical) TRUE if model has instantaneous absorption

has_linear_odes *has_linear_odes*

Description

Check if model has a linear ODE system

Usage

has_linear_odes(model)

Arguments

model (Model) PharmPy model

Value

(logical) TRUE if model has an ODE system that is linear

See Also

has_odes
has_linear_odes_with_real_eigenvalues

Examples

```
## Not run:  
model <- load_example_model("pheno")  
has_linear_odes(model)  
  
## End(Not run)
```

has_linear_odes_with_real_eigenvalues
has_linear_odes_with_real_eigenvalues

Description

Check if model has a linear ode system with real eigenvalues

Usage

```
has_linear_odes_with_real_eigenvalues(model)
```

Arguments

model (Model) PharmPy model

Value

(logical) TRUE if model has an ODE system that is linear

See Also

has_odes
has_linear_odes

Examples

```
## Not run:  
model <- load_example_model("pheno")  
has_linear_odes_with_real_eigenvalues(model)  
  
## End(Not run)
```

```
has_michaelis_menten_elimination  
    has_michaelis_menten_elimination
```

Description

Check if the model describes Michaelis-Menten elimination

This function relies on heuristics and will not be able to detect all possible ways of coding the Michaelis-Menten elimination.

Usage

```
has_michaelis_menten_elimination(model)
```

Arguments

model (Model) PharmPy model

Value

(logical) TRUE if model has describes Michaelis-Menten elimination

Examples

```
## Not run:  
model <- load_example_model("pheno")  
has_michaelis_menten_elimination(model)  
model <- set_michaelis_menten_elimination(model)  
has_michaelis_menten_elimination(model)  
  
## End(Not run)
```

```
has_mixed_mm_fo_elimination  
    has_mixed_mm_fo_elimination
```

Description

Check if the model describes mixed Michaelis-Menten and first order elimination

This function relies on heuristics and will not be able to detect all possible ways of coding the mixed Michaelis-Menten and first order elimination.

Usage

```
has_mixed_mm_fo_elimination(model)
```

Arguments

model (Model) PharmPy model

Value

(logical) TRUE if model has describes Michaelis-Menten elimination

Examples

```
## Not run:
model <- load_example_model("pheno")
has_mixed_mm_fo_elimination(model)
model <- set_mixed_mm_fo_elimination(model)
has_mixed_mm_fo_elimination(model)

## End(Not run)
```

has_mu_reference	<i>has_mu_reference</i>
------------------	-------------------------

Description

Check if model is Mu-reference or not.

Will return TRUE if each parameter with an ETA is dependent on a Mu parameter.

Usage

```
has_mu_reference(model)
```

Arguments

model (Model) PharmPy model

Value

(logical) Whether the model is mu referenced

has_odes	<i>has_odes</i>
----------	-----------------

Description

Check if model has an ODE system

Usage

```
has_odes(model)
```

Arguments

model (Model) PharmPy model

Value

(logical) TRUE if model has an ODE system

See Also

has_linear_odes
has_linear_odes_with_real_eigenvalues

Examples

```
## Not run:  
model <- load_example_model("pheno")  
has_odes(model)  
  
## End(Not run)
```

has_presystemic_metabolite	<i>has_presystemic_metabolite</i>
----------------------------	-----------------------------------

Description

Checks whether a model has a presystemic metabolite

If pre-systemic drug there will be a flow from DEPOT to METABOLITE as well as being a flow from the CENTRAL to METABOLITE

Usage

```
has_presystemic_metabolite(model)
```

Arguments

model (Model) Pharmpy model

Value

(logical) Whether a model has presystemic metabolite

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- add_metabolite(model, presystemic=TRUE)  
has_presystemic_metabolite(model)  
  
## End(Not run)
```

has_proportional_error_model
has_proportional_error_model

Description

Check if a model has a proportional error model

Multiple dependent variables are supported. By default the only (in case of one) or the first (in case of many) dependent variable is going to be checked.

Usage

```
has_proportional_error_model(model, dv = NULL)
```

Arguments

model (Model) Pharmpy model

dv (Expr or str or numeric (optional)) Name or DVID of dependent variable. NULL for the default (first or only)

Value

(logical) TRUE if the model has a proportional error model and FALSE otherwise

See Also

has_additive_error_model : Check if a model has an additive error model

has_combined_error_model : Check if a model has a combined error model

has_weighted_error_model : Check if a model has a weighted error model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
has_proportional_error_model(model)  
  
## End(Not run)
```

has_random_effect *has_random_effect*

Description

Decides whether the given parameter of a :class:pharmpy.model has a random effect.

Usage

```
has_random_effect(model, parameter, level = "all")
```

Arguments

model	(Model) Pharmpy model
parameter	(str) Input parameter
level	(str) The variability level to look for: 'iiv', 'iov', or 'all' (default)

Value

(logical) Whether the given parameter has a random effect

See Also

```
get_individual_parameters  
get_rv_parameters
```

Examples

```
## Not run:  
model <- load_example_model("pheno")  
has_random_effect(model, 'S1')  
has_random_effect(model, 'CL', 'iiv')  
has_random_effect(model, 'CL', 'iov')  
  
## End(Not run)
```

`has_seq_zo_fo_absorption`
has_seq_zo_fo_absorption

Description

Check if ode system describes a sequential zero-order, first-order absorption
Defined as the model having both zero- and first-order absorption.

Usage

`has_seq_zo_fo_absorption(model)`

Arguments

`model` (Model) PharmPy model

Value

(logical) TRUE if model has sequential zero-first absorption

See Also

`has_zero_order_absorption`
`has_first_order_absorption`

`has_weibull_absorption`
has_weibull_absorption

Description

Check if ode system describes a weibull type absorption

Usage

`has_weibull_absorption(model)`

Arguments

`model` (Model) PharmPy model

Value

(Bool : TRUE if model has weibull type absorption)

has_weighted_error_model
has_weighted_error_model

Description

Check if a model has a weighted error model

Usage

```
has_weighted_error_model(model)
```

Arguments

model (Model) PharmPy model

Value

(logical) TRUE if the model has a weighted error model and FALSE otherwise

See Also

has_additive_error_model : Check if a model has an additive error model

has_combined_error_model : Check if a model has a combined error model

has_proportional_error_model : Check if a model has a proportional error model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
has_weighted_error_model(model)  
  
## End(Not run)
```

has_zero_order_absorption
has_zero_order_absorption

Description

Check if ode system describes a zero order absorption
currently defined as having Infusion dose with rate not in dataset

Usage

```
has_zero_order_absorption(model)
```

Arguments

model (Model) Pharmpy model

Value

(logical) Whether the model has zero order absorption or not

Examples

```
## Not run:  
model <- load_example_model("pheno")  
has_zero_order_absorption(model)  
  
## End(Not run)
```

```
has_zero_order_elimination  
  has_zero_order_elimination
```

Description

Check if the model describes zero-order elimination

This function relies on heuristics and will not be able to detect all possible ways of coding the zero-order elimination.

Usage

```
has_zero_order_elimination(model)
```

Arguments

model (Model) Pharmpy model

Value

(logical) TRUE if model has describes zero order elimination

Examples

```
## Not run:  
model <- load_example_model("pheno")  
has_zero_order_elimination(model)  
model <- set_zero_order_elimination(model)  
has_zero_order_elimination(model)  
  
## End(Not run)
```

infer_datatypes	<i>infer_datatypes</i>
-----------------	------------------------

Description

Infer and set datatypes for the dataset

All columns or only the ones set in the columns option will be checked for conversion to a simpler datatype. Currently only int32 will be checked since this corresponds to the integer type in R.

Usage

```
infer_datatypes(model, columns = NULL)
```

Arguments

model	(Model) Pharmpy model
columns	(array(str) (optional)) The columns to attempt conversion or NULL for all

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- infer_datatypes(model, c('APGR'))
model$dataset

## End(Not run)
```

insert_ebes_into_dataset	<i>insert_ebes_into_dataset</i>
--------------------------	---------------------------------

Description

Insert EBEs and ETCs from results into the dataset of a model

Usage

```
insert_ebes_into_dataset(
  model,
  individual_estimates,
  individual_estimates_covariance = NULL
)
```

Arguments

model (Model) Pharmpy model
individual_estimates (data.frame) Individual eta estimates (EBEs). Could be taken directly from ModelfitResults.
individual_estimates_covariance (data.frame (optional)) Uncertainties of individual estimates (ETCs). Could be taken directly from ModelfitResults.

Value

(Model) Updated Pharmpy model

Examples

```

## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
ebes <- results$individual_estimates
etc <- results$individual_estimates_covariance
model2 <- insert_ebes_into_dataset(model, ebes, etc)
model2$datainfo$names

## End(Not run)

```

install_pharmpy *Install Pharmpy*

Description

Install the pharmpy-core python package into virtual environment. Uses the same Pharmpy version as pharmr.

Usage

```
install_pharmpy(envname = "r-reticulate", method = "auto")
```

Arguments

envname (str) name of environment. Default is r-reticulate
method (str) type of environment type (virtualenv, conda). Default is auto (virtualenv is not available on Windows)

install_pharmpy_devel *Install Pharmpy (with specified version)*

Description

Install the pharmpy-core python package into virtual environment.

Usage

```
install_pharmpy_devel(
    envname = "r-reticulate",
    method = "auto",
    version = "devel"
)
```

Arguments

envname	(str) name of environment. Default is r-reticulate
method	(str) type of environment type (virtualenv, conda). Default is auto (virtualenv is not available on Windows)
version	(str) which pharmpy version to use (use 'same' for most cases)

is_binary *is_binary*

Description

Check if an expression defines a binary variable (only having values 0 and 1)

Usage

```
is_binary(model, expr)
```

Arguments

model	(Model) Pharmpy model
expr	(str) A mathematical expression containing column or variable names

Value

(logical) TRUE if binary

Examples

```
## Not run:
model <- load_example_model("pheno")
is_binary(model, "WGT")
is_binary(model, "FA1")

## End(Not run)
```

<i>is_linearized</i>	<i>is_linearized</i>
----------------------	----------------------

Description

Determine if a model is linearized

Usage

```
is_linearized(model)
```

Arguments

model (Model) PharmPy model

Value

(logical) TRUE if model has been linearized and FALSE otherwise

Examples

```
## Not run:
model1 <- load_example_model("pheno")
is_linearized(model1)
model2 <- load_example_model("pheno_linear")
is_linearized(model2)

## End(Not run)
```

<code>is_real</code>	<code>is_real</code>
----------------------	----------------------

Description

Determine if an expression is real valued given constraints of a model

Usage

```
is_real(model, expr)
```

Arguments

<code>model</code>	(Model) Pharmpy model
<code>expr</code>	(numeric or str or Expr) Expression to test

Value

(logical or NULL) TRUE if expression is real, FALSE if not and NULL if unknown

Examples

```
## Not run:  
model <- load_example_model("pheno")  
is_real(model, "CL")  
  
## End(Not run)
```

<code>is_simulation_model</code>	<code>is_simulation_model</code>
----------------------------------	----------------------------------

Description

Check if a model is a pure simulation model

Usage

```
is_simulation_model(model)
```

Arguments

<code>model</code>	(Model) Pharmpy model
--------------------	-----------------------

Value

(logical) TRUE if it is a simulation model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
is_simulation_model(model)  
  
## End(Not run)
```

```
is_strictness_fulfilled  
      is_strictness_fulfilled
```

Description

Takes a ModelfitResults object and a statement as input and returns TRUE/FALSE if the evaluation of the statement is TRUE/FALSE.

Usage

```
is_strictness_fulfilled(model, results, strictness)
```

Arguments

model	(Model) Model for parameter specific strictness.
results	(ModelfitResults) ModelfitResults object
strictness	(str) A strictness expression

Value

(logical) A logical indicating whether the strictness criteria are fulfilled or not.

Examples

```
## Not run:  
res <- load_example_modelfit_results('pheno')  
model <- load_example_model('pheno')  
is_strictness_fulfilled(model, res, "minimization_successful or rounding_errors")  
  
## End(Not run)
```

<code>list_models</code>	<i>list_models</i>
--------------------------	--------------------

Description

List names of all models in a context

Will by default vector only models in the top level, but can vector all recursively using the recursive option. This will add the context path to each model name as a qualifier.

Usage

```
list_models(context, recursive = FALSE)
```

Arguments

<code>context</code>	(Context) The context
<code>recursive</code>	(logical) Only top level or all levels recursively down.

Value

(vector<str>) A vector of the model names

<code>list_time_varying_covariates</code>	<i>list_time_varying_covariates</i>
---	-------------------------------------

Description

Return a vector of names of all time varying covariates

Usage

```
list_time_varying_covariates(model)
```

Arguments

<code>model</code>	(Model) Pharmpy model
--------------------	-----------------------

Value

(vector) Names of all time varying covariates

See Also

`get_covariate_baselines` : get baselines for all covariates

Examples

```
## Not run:
model <- load_example_model("pheno")
list_time_varying_covariates(model)

## End(Not run)
```

load_dataset	<i>load_dataset</i>
--------------	---------------------

Description

Load the dataset given datainfo

Usage

```
load_dataset(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) Updated PharmPy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- unload_dataset(model)
model$dataset is NULL
model <- load_dataset(model)
model$dataset

## End(Not run)
```

```
load_example_model    load_example_model
```

Description

Load an example model

Load an example model from models built into PharmPy

Usage

```
load_example_model(name)
```

Arguments

name (str) Name of the model. Currently available models are "pheno" and "pheno_linear"

Value

(Model) PharmPy model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model$statements  
  
## End(Not run)
```

```
load_example_modelfit_results  
    load_example_modelfit_results
```

Description

Load the modelfit results of an example model

Load the modelfit results of an example model built into PharmPy

Usage

```
load_example_modelfit_results(name)
```

Arguments

name (str) Name of the model. Currently available models are "pheno" and "pheno_linear"

Value

(ModelfitResults) Loaded modelfit results object

Examples

```
## Not run:
results <- load_example_modelfit_results("pheno")
results$parameter_estimates

## End(Not run)
```

make_declarative	<i>make_declarative</i>
------------------	-------------------------

Description

Make the model statements declarative
Each symbol will only be declared once.

Usage

```
make_declarative(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) Updated PharmPy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model$statements$before_odes
model <- make_declarative(model)
model$statements$before_odes

## End(Not run)
```

map_eta_parameters *map_eta_parameters*

Description

Create a map with the connections from and to individual parameters, omegas and/or etas
The mapping will always be one to many.

Usage

```
map_eta_parameters(model, keys, values, level = "iiv")
```

Arguments

model	(Model) Pharmpy model
keys	(str) What to map from. Either parameters, omegas or etas
values	(str) What to map to. Either parameters, omegas or etas
level	(str) Which variability level to consider. Either iiv or iov. iiv is the default.

Value

(list) A list from parameters, omegas or etas to a vector of connected parameters, omegas or etas.

Examples

```
## Not run:
model <- load_example_model("pheno")
map_eta_parameters(model, "parameters", "omegas")

## End(Not run)
```

mu_reference_model *mu_reference_model*

Description

Convert model to use mu-referencing

Mu-referencing an eta is to separately define its actual mu (mean) parameter. For example: (equation could not be rendered, see API doc on website) normal distribution would give (equation could not be rendered, see API doc on website) (equation could not be rendered, see API doc on website)

Usage

```
mu_reference_model(model)
```

Arguments

model (Model) Pharmpy model

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- mu_reference_model(model)
model$statements$before_odes

## End(Not run)
```

omit_data

omit_data

Description

Iterate over omissions of a certain group in a dataset. One group is omitted at a time.

Usage

```
omit_data(dataset_or_model, group, name_pattern = "omitted_{}")
```

Arguments

dataset_or_model (data.frame or Model) Dataset or model for which to omit records

group (str) Name of the column to use for grouping

name_pattern (str) Name to use for generated datasets. A number starting from 1 will be put in the placeholder.

Value

(iterator) Iterator yielding tuples of models/dataframes and the omitted group

open_context *open_context*

Description

Open a context from a tool run

Usage

```
open_context(name, ref = NULL)
```

Arguments

name (str) Name of the context
ref (str (optional)) Parent path of the context

Examples

```
## Not run:  
ctx <- open_context("myrun")  
  
## End(Not run)
```

plot_abs_cwres_vs_ipred
 plot_abs_cwres_vs_ipred

Description

Plot \CWRES\ vs IPRED

Usage

```
plot_abs_cwres_vs_ipred(  
  model,  
  predictions,  
  residuals,  
  stratify_on = NULL,  
  bins = 8  
)
```

Arguments

model	(Model) Pharmpy model
predictions	(data.frame) DataFrame containing the predictions
residuals	(data.frame) DataFrame containing the residuals
stratify_on	(str (optional)) Name of parameter for stratification
bins	(numeric) Number of bins for stratification

Value

(alt.Chart) Plot

Examples

```
## Not run:
model <- load_example_model("pheno")
res <- load_example_modelfit_results("pheno")
plot_abs_cwres_vs_ipred(model, res$predictions, res$residuals)
model <- load_example_model("pheno")
res <- load_example_modelfit_results("pheno")
plot_abs_cwres_vs_ipred(model, res$predictions, res$residuals, 'WGT', bins=4)

## End(Not run)
```

plot_cwres_vs_idv *plot_cwres_vs_idv*

Description

Plot CWRES vs idv

Usage

```
plot_cwres_vs_idv(model, residuals, stratify_on = NULL, bins = 8)
```

Arguments

model	(Model) Pharmpy model
residuals	(data.frame) DataFrame containing CWRES
stratify_on	(str (optional)) Name of parameter for stratification
bins	(numeric) Number of bins for stratification

Value

(alt.Chart) Plot

Examples

```
## Not run:
model <- load_example_model("pheno")
res <- load_example_modelfit_results("pheno")
plot_cwres_vs_idv(model, res$residuals)
model <- load_example_model("pheno")
res <- load_example_modelfit_results("pheno")
plot_cwres_vs_idv(model, res$residuals, 'WGT', bins=4)

## End(Not run)
```

plot_dv_vs_ipred *plot_dv_vs_ipred*

Description

Plot DV vs IPRED

Usage

```
plot_dv_vs_ipred(model, predictions, stratify_on = NULL, bins = 8)
```

Arguments

model	(Model) PharmPy model
predictions	(data.frame) DataFrame containing the predictions
stratify_on	(str (optional)) Name of parameter for stratification
bins	(numeric) Number of bins for stratification

Value

(alt.Chart) Plot

Examples

```
## Not run:
model <- load_example_model("pheno")
res <- load_example_modelfit_results("pheno")
plot_dv_vs_ipred(model, res$predictions)
model <- load_example_model("pheno")
res <- load_example_modelfit_results("pheno")
plot_dv_vs_ipred(model, res$predictions, 'WGT', bins=4)

## End(Not run)
```

plot_dv_vs_pred *plot_dv_vs_pred*

Description

Plot DV vs PRED

Usage

```
plot_dv_vs_pred(model, predictions, stratify_on = NULL, bins = 8)
```

Arguments

model	(Model) Pharmpy model
predictions	(data.frame) DataFrame containing the predictions
stratify_on	(str (optional)) Name of parameter for stratification
bins	(numeric) Number of bins for stratification

Value

(alt.Chart) Plot

Examples

```
## Not run:
model <- load_example_model("pheno")
res <- load_example_modelfit_results("pheno")
plot_dv_vs_pred(model, res$predictions)
model <- load_example_model("pheno")
res <- load_example_modelfit_results("pheno")
plot_dv_vs_pred(model, res$predictions, 'WGT', bins=4)

## End(Not run)
```

plot_eta_distributions
plot_eta_distributions

Description

Plot eta distributions for all etas

Usage

```
plot_eta_distributions(model, individual_estimates)
```

Arguments

`model` (Model) Pharmpy model
`individual_estimates` (data.frame) Individual estimates for etas

Value

(alt.Chart) Plot

Examples

```
## Not run:
model <- load_example_model("pheno")
res <- load_example_modelfit_results("pheno")
plot_eta_distributions(model, res$individual_estimates)

## End(Not run)
```

`plot_individual_predictions`
plot_individual_predictions

Description

Plot DV and predictions grouped on individuals

The predictions would normally be taken from the modelfit results, but any data frame of appropriate format can be used. It should have one column per type of prediction where the column name is the type (e.g. "PRED" or "IPRED"), one row for each record of the dataset in the model. A predictions table containing only some of the individuals is ok to use, but then care needs to be taken to get match the row index of the original dataset.

Usage

```
plot_individual_predictions(model, predictions, individuals = NULL)
```

Arguments

`model` (Model) Pharmpy model
`predictions` (data.frame) One column for each type of prediction
`individuals` (array(numeric) (optional)) A vector of individuals to include. NULL for all individuals

Value

(alt.Chart) Plot

Examples

```
## Not run:
model <- load_example_model("pheno")
res <- load_example_modelfit_results("pheno")
plot_individual_predictions(model, res$predictions, individuals=c(1, 2, 3, 4, 5))

## End(Not run)
```

plot_iofv_vs_iofv *plot_iofv_vs_iofv*

Description

Plot individual OFV of two models against each other

Usage

```
plot_iofv_vs_iofv(iofv1, iofv2, name1, name2)
```

Arguments

iofv1	(array) Estimated iOFV of the first model
iofv2	(array) Estimated iOFV of the second model
name1	(str) Name of first model
name2	(str) Name of second model

Value

(alt.Chart) Scatterplot

Examples

```
## Not run:
res1 <- load_example_modelfit_results("pheno")
res2 <- load_example_modelfit_results("pheno_linear")
plot_iofv_vs_iofv(res1$individual_ofv, res2$individual_ofv, "nonlin", "linear")

## End(Not run)
```

plot_transformed_eta_distributions
plot_transformed_eta_distributions

Description

Plot transformed eta distributions for all transformed etas

Usage

```
plot_transformed_eta_distributions(  
  model,  
  parameter_estimates,  
  individual_estimates,  
  parameter_estimates_untransformed,  
  seed = 1234  
)
```

Arguments

model (Model) PharmPy model
parameter_estimates (array or list(str=numeric)) Parameter estimates of model fit
individual_estimates (data.frame) Individual estimates for etas
parameter_estimates_untransformed (array or list(str=numeric)) Parameter estimates of untransformed model fit
seed (numeric) Random number generator or seed

Value

(alt.Chart) Plot

plot_vpc *plot_vpc*

Description

Creates a VPC plot for a model

Usage

```
plot_vpc(
  model,
  simulations,
  binning = "equal_number",
  nbins = 8,
  qi = 0.95,
  ci = 0.95,
  stratify_on = NULL
)
```

Arguments

model	(Model) Pharmpy model
simulations	(str or data.frame) DataFrame containing the simulation data or path to dataset. The dataset has to have one (index) column named "SIM" containing the simulation number, one (index) column named "index" containing the data indices and one dv column. See below for more information.
binning	(str) Binning method. Can be "equal_number" or "equal_width". The default is "equal_number".
nbins	(numeric) Number of bins. Default is 8.
qi	(numeric) Upper quantile. Default is 0.95.
ci	(numeric) Confidence interval. Default is 0.95.
stratify_on	(str (optional)) Parameter to use for stratification. Optional.

Value

(alt.Chart) Plot The simulation data should have the following format: +---+---+---+ | SIM
| index | DV | +=====+=====+=====+ | 1 | 0 | 0.000 | +---+---+---+ | 1 | 1 | 34.080 |
+---+---+---+ | 1 | 2 | 28.858 | +---+---+---+ | 1 | 3 | 0.000 | +---+---+---+ | 1
| 4 | 12.157 | +---+---+---+ | 2 | 0 | 23.834 | +---+---+---+ | 2 | 1 | 0.000 | +---+---
-+---+---+ | ... | ... | ... | +---+---+---+ | 20 | 2 | 0.000 | +---+---+---+ | 20 | 3 | 31.342 |
+---+---+---+ | 20 | 4 | 29.983 | +---+---+---+

Examples

```
## Not run:
model <- load_example_model("pheno")
sim_model <- set_simulation(model, n=100)
sim_data <- run_simulation(sim_model)
plot_vpc(model, sim_data)

## End(Not run)
```

predict_influential_individuals
predict_influential_individuals

Description

Predict influential individuals for a model using a machine learning model.

Please refer to www.page-meeting.org/?abstract=10029 for more information on training and estimated precision and accuracy.

Usage

```
predict_influential_individuals(model, results, cutoff = 3.84)
```

Arguments

model	(Model) PharmPy model
results	(ModelFitResults) Results for model
cutoff	(numeric) Cutoff threshold for a dofV signalling an influential individual

Value

(data.frame) Dataframe over the individuals with a dofV column containing the raw predicted delta-OFV and an influential column with a boolean to tell whether the individual is influential or not.

See Also

predict_influential_outliers
 predict_outliers

predict_influential_outliers
predict_influential_outliers

Description

Predict influential outliers for a model using a machine learning model.

Please refer to www.page-meeting.org/?abstract=10029 for more information on training and estimated precision and accuracy.

Usage

```
predict_influential_outliers(
  model,
  results,
  outlier_cutoff = 3,
  influential_cutoff = 3.84
)
```

Arguments

model (Model) Pharmpy model

results (ModelfitResults) Results for model

outlier_cutoff (numeric) Cutoff threshold for a residual signaling an outlier

influential_cutoff (numeric) Cutoff threshold for a dofv signaling an influential individual

Value

(data.frame) Dataframe over the individuals with a outliers and dofv columns containing the raw predictions and influential, outlier and influential_outlier boolean columns.

See Also

predict_influential_individuals

predict_outliers

predict_outliers *predict_outliers*

Description

Predict outliers for a model using a machine learning model.

See the :ref:simeval <Individual OFV summary> documentation for a definition of the residual

Please refer to www.page-meeting.org/?abstract=10029 for more information on training and estimated precision and accuracy.

Usage

```
predict_outliers(model, results, cutoff = 3)
```

Arguments

model (Model) Pharmpy model

results (ModelfitResults) ModelfitResults for the model

cutoff (numeric) Cutoff threshold for a residual signaling an outlier

Value

(data.frame) Dataframe over the individuals with a residual column containing the raw predicted residuals and a outlier column with a boolean to tell whether the individual is an outlier or not.

See Also

`predict_influential_individuals`

`predict_influential_outliers`

Examples

```
## Not run:  
model <- load_example_model("pheno")  
results <- load_example_modelfit_results("pheno")  
predict_outliers(model, results)  
  
## End(Not run)
```

`print_fit_summary` *print_fit_summary*

Description

Print a summary of the model fit

Usage

```
print_fit_summary(model, modelfit_results)
```

Arguments

`model` (Model) PharmPy model

`modelfit_results`
 (ModelfitResults) PharmPy ModelfitResults object

print_log	<i>print_log</i>
-----------	------------------

Description

Print the log of a context

Usage

```
print_log(context)
```

Arguments

context	(Context) Print the log of this context
---------	---

print_model_code	<i>print_model_code</i>
------------------	-------------------------

Description

Print the model code of the underlying model language to the console

Usage

```
print_model_code(model)
```

Arguments

model	(Model) Pharmpy model
-------	-----------------------

Examples

```
## Not run:  
model <- load_example_model("pheno")  
print_model_code(model)  
  
## End(Not run)
```

`print_model_symbols` *print_model_symbols*

Description

Print all symbols defined in a model

Symbols will be in one of the categories thetas, etas, omegas, epsilons, sigmas, variables and data columns

Usage

```
print_model_symbols(model)
```

Arguments

`model` (Model) Pharmpy model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
print_model_symbols(model)  
  
## End(Not run)
```

`print_pharmpy_version` *Print pharmpy version*

Description

Print the pharmpy version pharmr uses.

Usage

```
print_pharmpy_version()
```

```
read_dataset_from_datainfo
    read_dataset_from_datainfo
```

Description

Read a dataset given a datainfo object or path to a datainfo file

Usage

```
read_dataset_from_datainfo(datainfo, datatype = NULL)
```

Arguments

datainfo (DataInfo or str) A datainfo object or a path to a datainfo object
 datatype (str (optional)) A string to specify dataset type

Value

(data.frame) The dataset

```
read_model          read_model
```

Description

Read model from file

Usage

```
read_model(path, missing_data_token = NULL)
```

Arguments

path (str) Path to model
 missing_data_token (str (optional)) Use this token for missing data. This option will override the token from the config. (This option was added in PharmPy version 1.2.0)

Value

(Model) PharmPy model

See Also

`read_model_from_database` : Read model from database
`read_model_from_string` : Read model from string

Examples

```
## Not run:
model <- read_model("/home/run1$mod")

## End(Not run)
```

`read_modelfit_results` *read_modelfit_results*

Description

Read results from external tool for a model

Usage

```
read_modelfit_results(path, esttool = NULL)
```

Arguments

`path` (str) Path to model file
`esttool` (str) Set if other than the default estimation tool is to be used

Value

(ModelfitResults) Results object

`read_model_from_string`
read_model_from_string

Description

Read model from the model code in a string

Usage

```
read_model_from_string(code)
```

Arguments

`code` (str) Model code to read

Value

(Model) Pharmpy model

See Also

read_model : Read model from file

read_model_from_database : Read model from database

Examples

```
## Not run:  
s <- "$PROBLEM  
$INPUT ID DV TIME  
$DATA file$csv  
$PRED  
Y=THETA(1)+ETA(1)+ERR(1)  
$THETA 1  
$OMEGA 0.1  
$SIGMA 1  
$ESTIMATION METHOD=1"  
read_model_from_string(s)  
  
## End(Not run)
```

read_results

read_results

Description

Read results object from file

Usage

```
read_results(path)
```

Arguments

path (str) Path to results file

Value

(Results) Results object for tool

See Also

create_results

Examples

```
## Not run:  
res <- read_results("results$json")  
  
## End(Not run)
```

```
remove_bioavailability  
      remove_bioavailability
```

Description

Remove bioavailability from the first dose compartment of model.

Usage

```
remove_bioavailability(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) Updated PharmPy model

See Also

set_bioavailability

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- remove_bioavailability(model)  
  
## End(Not run)
```

```
remove_covariate_effect
      remove_covariate_effect
```

Description

Remove a covariate effect from an instance of :class:pharmpy.model.

Usage

```
remove_covariate_effect(model, parameter, covariate, keep_fixed = TRUE)
```

Arguments

model	(Model) Pharmpy model
parameter	(str) Name of parameter.
covariate	(str) Name of covariate.
keep_fixed	(logical) Whether to keep fixed effects (e.g. allometry) or not. Default is TRUE

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:
model <- load_example_model("pheno")
has_covariate_effect(model, "CL", "WGT")
model <- remove_covariate_effect(model, "CL", "WGT", keep_fixed=FALSE)
has_covariate_effect(model, "CL", "WGT")

## End(Not run)
```

```
remove_derivative      remove_derivative
```

Description

Remove a derivative currently being calculate when running model. Currently, only derivatives with respect to the prediction is supported. Default is to remove all that are present, First order derivatives are specied either by single string or single-element tuple. For instance with_respect_to = "ETA_1" or with_respect_to = ("ETA_1",)

Second order derivatives are specified by giving the two independent variables in a tuple of tuples. For instance with_respect_to ((ETA_1, EPS_1),)

Multiple derivatives can be specified within a tuple. For instance ((ETA_1, EPS_1), "ETA_1")

Currently, only ETAs and EPSILONS are supported

Usage

```
remove_derivative(model, with_respect_to = NULL)
```

Arguments

`model` (Model) Pharmpy model
`with_respect_to` (array(array(str) or str) or str (optional)) Parameter name(s) to use as independent variables. Default is NULL.

Value

(Model) Updated Pharmpy model

<code>remove_error_model</code>	<i>remove_error_model</i>
---------------------------------	---------------------------

Description

Remove error model.

Usage

```
remove_error_model(model)
```

Arguments

`model` (Model) Pharmpy model

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model$statements$find_assignment("Y")  
model <- remove_error_model(model)  
model$statements$find_assignment("Y")  
  
## End(Not run)
```

```
remove_estimation_step  
    remove_estimation_step
```

Description

Remove estimation step

Usage

```
remove_estimation_step(model, idx)
```

Arguments

model	(Model) PharmPy model
idx	(numeric) index of estimation step to remove (starting from 0)

Value

(Model) Update PharmPy model

See Also

```
add_estimation_step  
set_estimation_step  
append_estimation_step_options  
add_parameter_uncertainty_step  
remove_parameter_uncertainty_step  
set_evaluation_step
```

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- remove_estimation_step(model, 0)  
ests <- model$execution_steps  
length(ests)  
  
## End(Not run)
```

`remove_iiv`*remove_iiv*

Description

Removes all IIV etas given a vector with eta names and/or parameter names.

Usage

```
remove_iiv(model, to_remove = NULL)
```

Arguments

<code>model</code>	(Model) Pharmpy model
<code>to_remove</code>	(array(str) or str (optional)) Name/names of etas and/or name/names of individual parameters to remove. If NULL, all etas that are IIVs will be removed. NULL is default.

Value

(Model) Updated Pharmpy model

See Also

`remove_iov`

`add_iiv`

`add_iov`

`add_pk_iiv`

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- remove_iiv(model)
model$statements$find_assignment("CL")
model <- load_example_model("pheno")
model <- remove_iiv(model, "VC")
model$statements$find_assignment("VC")

## End(Not run)
```

remove_iov	<i>remove_iov</i>
------------	-------------------

Description

Removes all IOV etas given a vector with eta names.

Usage

```
remove_iov(model, to_remove = NULL)
```

Arguments

model	(Model) Pharmpy model
to_remove	(array(str) or str (optional)) Name/names of IOV etas to remove, e.g. 'ETA_IOV_1_1'. If NULL, all etas that are IOVs will be removed. NULL is default.

Value

(Model) Updated Pharmpy model

See Also

add_iiv

add_iov

remove_iiv

add_pk_iiv

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- remove_iov(model)  
  
## End(Not run)
```

remove_lag_time *remove_lag_time*

Description

Remove lag time from the dose compartment of model.

Usage

```
remove_lag_time(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) Updated PharmPy model

See Also

set_transit_compartments
add_lag_time

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- remove_lag_time(model)  
  
## End(Not run)
```

remove_loq_data *remove_loq_data*

Description

Remove loq data records from the dataset
Does nothing if none of the limits are specified.

Usage

```
remove_loq_data(  
  model,  
  lloq = NULL,  
  uloq = NULL,  
  blq = NULL,  
  alq = NULL,  
  keep = 0  
)
```

Arguments

model	(Model) Pharmpy model
lloq	(numeric or str (optional)) Value or column name for lower limit of quantification.
uloq	(numeric or str (optional)) Value or column name for upper limit of quantification.
blq	(str (optional)) Column name for below limit of quantification indicator.
alq	(str (optional)) Column name for above limit of quantification indicator.
keep	(numeric) Number of loq records to keep for each run of consecutive loq records.

Value

(Model) Updated Pharmpy model

See Also

set_lloq_data
transform_blq

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- remove_loq_data(model, lloq=10, uloq=40)  
length(model$dataset)  
  
## End(Not run)
```

```
remove_parameter_uncertainty_step  
    remove_parameter_uncertainty_step
```

Description

Removes parameter uncertainty step from the final estimation step

Usage

```
remove_parameter_uncertainty_step(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) PharmPy model object

See Also

```
add_estimation_step  
set_estimation_step  
remove_estimation_step  
append_estimation_step_options  
add_parameter_uncertainty_step  
set_evaluation_step
```

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- remove_parameter_uncertainty_step(model)  
ests <- model$execution_steps  
ests[1]  
  
## End(Not run)
```

```
remove_peripheral_compartment
      remove_peripheral_compartment
```

Description

Remove a peripheral distribution compartment from model

If name is set, a peripheral compartment will be removed from the compartment with the specified name.

Initial estimates:

```
=====  
2 (equation could not be rendered, see API doc on website) 3 (equation could not be rendered, see  
API doc on website) =====
```

Usage

```
remove_peripheral_compartment(model, name = NULL)
```

Arguments

model	(Model) Pharmpy model
name	(str (optional)) Name of compartment to remove peripheral compartment from.

Value

(Model) Updated Pharmpy model

See Also

```
set_peripheral_compartment
add_peripheral_compartment
```

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_peripheral_compartments(model, 2)
model <- remove_peripheral_compartment(model)
model$statements$ode_system

## End(Not run)
```

remove_predictions *remove_predictions*

Description

Remove predictions and/or residuals

Remove predictions from estimation step. Default is to remove all predictions.

Usage

```
remove_predictions(model, to_remove = NULL)
```

Arguments

model (Model) Pharmpy model
to_remove (array(str) (optional)) Predictions to remove

Value

(Model) Updated Pharmpy model

See Also

add_predictions
add_residuals
set_estimation_step
add_estimation_step
remove_estimation_step
append_estimation_step_options
add_parameter_uncertainty_step
remove_parameter_uncertainty_step

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- remove_predictions(model)  
model$execution_steps[-1].predictions  
  
## End(Not run)
```

remove_residuals	<i>remove_residuals</i>
------------------	-------------------------

Description

Remove residuals

Remove residuals from estimation step. Default is to remove all residuals.

Usage

```
remove_residuals(model, to_remove = NULL)
```

Arguments

model	(Model) Pharmpy model
to_remove	(array(str) (optional)) Residuals to remove

Value

(Model) Updated Pharmpy model

See Also

add_predictions
add_residuals
set_estimation_step
add_estimation_step
remove_estimation_step
append_estimation_step_options
add_parameter_uncertainty_step
remove_parameter_uncertainty_step

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- remove_residuals(model)  
model$execution_steps[-1].residuals  
  
## End(Not run)
```

remove_unused_columns *remove_unused_columns*

Description

Remove all columns in the dataset that are not used by the model including dropped columns

Warnings Currently columns not needed to give a prediction, but used by other expressions in the model are kept. This will change in the future.

Usage

```
remove_unused_columns(model)
```

Arguments

model (Model) Pharmpy model

Value

(Model) Updated Pharmpy model

See Also

drop_columns : Drop columns from the dataset

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- remove_unused_columns(model)  
vector(model$dataset$columns)  
  
## End(Not run)
```

remove_unused_parameters_and_rvs
remove_unused_parameters_and_rvs

Description

Remove any parameters and rvs that are not used in the model statements

Usage

```
remove_unused_parameters_and_rvs(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) Updated PharmPy model

rename_symbols	<i>rename_symbols</i>
----------------	-----------------------

Description

Rename symbols in the model
Make sure that no name clash occur.

Usage

```
rename_symbols(model, new_names)
```

Arguments

model (Model) PharmPy model
new_names (list(str or Expr=str or Expr)) From old name or symbol to new name or symbol

Value

(Model) Updated PharmPy model

replace_fixed_thetas	<i>replace_fixed_thetas</i>
----------------------	-----------------------------

Description

Replace all fixed thetas with constants in the model statements

Usage

```
replace_fixed_thetas(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) Updated PharmPy model

```
replace_non_random_rvs
    replace_non_random_rvs
```

Description

Replace all random variables that are not actually random

Some random variables are constant. For example a normal distribution with the variance parameter fixed to 0 will always yield a single value when sampled. This function will find all such random variables and replace them with their constant value in the model.

Usage

```
replace_non_random_rvs(model)
```

Arguments

model (Model) Pharmpy model

Value

(Model) Updated Pharmpy model

```
resample_data    resample_data
```

Description

Iterate over resamples of a dataset.

The dataset will be grouped on the group column then groups will be selected randomly with or without replacement to form a new dataset. The groups will be renumbered from 1 and upwards to keep them separated in the new dataset.

Usage

```
resample_data(
    dataset_or_model,
    group,
    resamples = 1,
    stratify = NULL,
    sample_size = NULL,
    replace = FALSE,
    name_pattern = "resample_{}",
    name = NULL
)
```

Arguments

dataset_or_model	(data.frame or Model) Dataset or PharmPy model to use
group	(str) Name of column to group by
resamples	(numeric) Number of resamples (iterations) to make
stratify	(str (optional)) Name of column to use for stratification. The values in the stratification column must be equal within a group so that the group can be uniquely determined. A ValueError exception will be raised otherwise.
sample_size	(numeric (optional)) The number of groups that should be sampled. The default is the number of groups. If using stratification the default is to sample using the proportion of the strata in the dataset. A list of specific sample sizes for each stratum can also be supplied.
replace	(logical) A boolean controlling whether sampling should be done with or without replacement
name_pattern	(str) Name to use for generated datasets. A number starting from 1 will be put in the placeholder.
name	(str (optional)) Option to name pattern in case of only one resample

Value

(iterator) An iterator yielding tuples of a resampled DataFrame and a vector of resampled groups in order

reset_index	<i>Reset index</i>
-------------	--------------------

Description

Reset index of dataframe.

Reset index from a multi indexed data.frame so that index is added as columns

Usage

```
reset_index(df)
```

Arguments

df	A data.frame converted from python using reticulate
----	---

reset_indices_results *Reset result indices*

Description

Resets indices in dataframes within Results-objects when needed

Usage

```
reset_indices_results(res)
```

Arguments

res A Pharmpy results object

retrieve_model *retrieve_model*

Description

Retrieve a model from a context

Any models created and run by the tool can be retrieved.

Usage

```
retrieve_model(context, name)
```

Arguments

context (Context) A previously opened context

name (str) Name of the model or a qualified name with a subcontext path, e.g.

Value

(Model) Pharmpy model

Examples

```
## Not run:  
context <- open_context(ref='path/to/', name='modelsearch1')  
model <- retrieve_model(context, 'run1')  
  
## End(Not run)
```

```
retrieve_modelfit_results
      retrieve_modelfit_results
```

Description

Retrieve the modelfit results of a model

Usage

```
retrieve_modelfit_results(context, name)
```

Arguments

context	(Context) A previously opened context
name	(str) Name of the model or a qualified name with a subcontext path, e.g.

Value

(ModelfitResults) The results object

Examples

```
## Not run:
context <- open_context("iivsearch1")
results <- retrieve_modelfit_results(context, 'input')

## End(Not run)
```

```
retrieve_models      retrieve_models
```

Description

Retrieve models after a tool run
Any models created and run by the tool can be retrieved.

Usage

```
retrieve_models(source, names = NULL)
```

Arguments

source	(str or Context) Source where to find models. Can be a path (as str or Path), or a Context
names	(array(str) (optional)) List of names of the models to retrieve or NULL for all

Value

(vector) List of retrieved PharmPy models

Examples

```
## Not run:
tooldir_path <- 'path/to/tool/directory'
models <- retrieve_models(tooldir_path, names=c('run1'))

## End(Not run)
```

run_allometry

run_allometry

Description

Run allometry tool. For more details, see [:ref:allometry](#).

Usage

```
run_allometry(
  model,
  results,
  allometric_variable = "WT",
  reference_value = 70,
  parameters = NULL,
  initials = NULL,
  lower_bounds = NULL,
  upper_bounds = NULL,
  fixed = TRUE,
  ...
)
```

Arguments

model	(Model) PharmPy model
results	(ModelfitResults) Results for model
allometric_variable	(str or Expr) Name of the variable to use for allometric scaling (default is WT)
reference_value	(str or numeric or Expr) Reference value for the allometric variable (default is 70)
parameters	(array(str or Expr) (optional)) Parameters to apply scaling to (default is all CL, Q and V parameters)

initials	(array(numeric) (optional)) Initial estimates for the exponents. (default is to use 0.75 for CL and Qs and 1 for Vs)
lower_bounds	(array(numeric) (optional)) Lower bounds for the exponents. (default is 0 for all parameters)
upper_bounds	(array(numeric) (optional)) Upper bounds for the exponents. (default is 2 for all parameters)
fixed	(logical) Should the exponents be fixed or not. (default TRUE)
...	Arguments to pass to tool

Value

(AllometryResults) Allometry tool result object

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
run_allometry(model=model, results=results, allometric_variable='WGT')

## End(Not run)
```

run_amd

run_amd

Description

Run Automatic Model Development (AMD) tool

Usage

```
run_amd(
  input,
  results = NULL,
  modeltype = "basic_pk",
  administration = "oral",
  strategy = "default",
  cl_init = NULL,
  vc_init = NULL,
  mat_init = NULL,
  b_init = NULL,
  emax_init = NULL,
  ec50_init = NULL,
  met_init = NULL,
  search_space = NULL,
  lloq_method = NULL,
```

```

lloq_limit = NULL,
allometric_variable = NULL,
occasion = NULL,
strictness = "minimization_successful or (rounding_errors and sigdigs>=0.1)",
dv_types = NULL,
mechanistic_covariates = NULL,
retries_strategy = "all_final",
parameter_uncertainty_method = NULL,
ignore_datainfo_fallback = FALSE,
.E = NULL,
...
)

```

Arguments

input	(Model or str) Starting model or dataset
results	(ModelfitResults (optional)) Results of input if input is a model
modeltype	(str) Type of model to build. Valid strings are 'basic_pk', 'pkpd', 'drug_metabolite' and 'tmdd'
administration	(str) Route of administration. Either 'iv', 'oral' or 'ivoral'
strategy	(str) Run algorithm for AMD procedure. Valid options are 'default', 'reevaluation', 'SIR', 'SRI', and 'RSI'.
cl_init	(numeric (optional)) Initial estimate for the population clearance
vc_init	(numeric (optional)) Initial estimate for the central compartment population volume
mat_init	(numeric (optional)) Initial estimate for the mean absorption time (not for iv models)
b_init	(numeric (optional)) Initial estimate for the baseline (PKPD model)
emax_init	(numeric (optional)) Initial estimate for E_max (PKPD model)
ec50_init	(numeric (optional)) Initial estimate for EC_50 (PKPD model)
met_init	(numeric (optional)) Initial estimate for mean equilibration time (PKPD model)
search_space	(str (optional)) MFL for search space for structural and covariate model
lloq_method	(str (optional)) Method for how to remove LOQ data. See transform_bllq for vector of available methods
lloq_limit	(numeric (optional)) Lower limit of quantification. If NULL LLOQ column from dataset will be used
allometric_variable	(str or Expr (optional)) Variable to use for allometry. This option is deprecated. Please use ALLOMETRY in the mfl instead.
occasion	(str (optional)) Name of occasion column
strictness	(str) Strictness criteria
dv_types	(list(str=numeric) (optional)) Dictionary of DV types for TMDD models with multiple DVs.

mechanistic_covariates	(array(str or list(str)) (optional)) List of covariates or tuple of covariate and parameter combination to run in a separate proioritized covsearch run. For instance c("WT", ("CRCL", "CL")). The effects are extracted from the search space for covsearch.
retries_strategy	(str) Whether or not to run retries tool. Valid options are 'skip', 'all_final' or 'final'. Default is 'final'.
parameter_uncertainty_method	(str (optional)) Parameter uncertainty method.
ignore_datainfo_fallback	(logical) Ignore using datainfo to get information not given by the user. Default is FALSE
.E	(list(str=numeric or str) (optional)) EXPERIMENTAL FEATURE. Dictionary of different E-values used in mBIC
...	Arguments to pass to tool

Value

(AMDResults) Results for the run

See Also

run_iiv

run_tool

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
res <- run_amd(model, results=results)

## End(Not run)
```

run_bootstrap

run_bootstrap

Description

Run bootstrap tool

Usage

```
run_bootstrap(  
  model,  
  results = NULL,  
  samples = 1,  
  dof = FALSE,  
  strictness = "",  
  ...  
)
```

Arguments

model	(Model) PharmPy model
results	(ModelFitResults (optional)) Results for model
samples	(numeric) Number of bootstrap resamples
dof	(logical) Will evaluate bootstrap models with original dataset if set
strictness	(str) Strictness expression for which models should be used in the calculations. Default is all models
...	Arguments to pass to tool

Value

(BootstrapResults) Bootstrap tool result object

Examples

```
## Not run:  
model <- load_example_model("pheno")  
results <- load_example_model_fit_results("pheno")  
run_bootstrap(model, res, samples=500)  
  
## End(Not run)
```

run_covsearch

run_covsearch

Description

Run COVsearch tool. For more details, see :ref:covsearch.

Usage

```
run_covsearch(
  model,
  results,
  search_space,
  p_forward = 0.01,
  p_backward = 0.001,
  max_steps = -1,
  algorithm = "scm-forward-then-backward",
  max_eval = FALSE,
  adaptive_scope_reduction = FALSE,
  strictness = "minimization_successful or (rounding_errors and sigdigs>=0.1)",
  parameter_uncertainty_method = NULL,
  naming_index_offset = 0,
  nsamples = 10,
  .samba_max_covariates = 3,
  .samba_selection_criterion = "bic",
  .samba_linreg_method = "ols",
  .samba_stepwise_lcs = NULL,
  ...
)
```

Arguments

model	(Model) Pharmpy model
results	(ModelfitResults) Results of model
search_space	(str or ModelFeatures) MFL of covariate effects to try
p_forward	(numeric) The p-value to use in the likelihood ratio test for forward steps
p_backward	(numeric) The p-value to use in the likelihood ratio test for backward steps
max_steps	(numeric) The maximum number of search steps to make
algorithm	(str) The search algorithm to use. Currently, 'scm-forward' and 'scm-forward-then-backward' are supported.
max_eval	(logical) Limit the number of function evaluations to 3.1 times that of the base model. Default is FALSE.
adaptive_scope_reduction	(logical) Stash all non-significant parameter-covariate effects to be tested after all significant effects have been tested. Once all these have been tested, try adding the stashed effects once more with a regular forward approach. Default is FALSE
strictness	(str) Strictness criteria
parameter_uncertainty_method	(str (optional)) Parameter uncertainty method. Will be used in ranking models if strictness includes parameter uncertainty
naming_index_offset	(numeric (optional)) index offset for naming of runs. Default is 0.

nsamples (numeric) Number of samples from individual parameter conditional distribution for linear covariate model selection. Default is 10, i.e. generating 10 samples per subject
 .samba_max_covariates (numeric (optional)) Maximum number of covariate inclusion allowed in linear covariate screening for each parameter.
 .samba_selection_criterion (str) Method used to fit linear covariate models. Currently, Ordinary Least Squares (ols), Weighted Least Squares (wls), and Linear Mixed-Effects (lme) are supported.
 .samba_linreg_method (str) Method used for linear and nonlinear model selection in SAMBA methods. Currently, BIC and LRT are supported.
 .samba_stepwise_lcs (logical (optional)) Use stepwise linear covariate screening or not. By default, SAMBA methods use stepwise LCS whereas SCM-LCS uses non-stepwise LCS
 ... Arguments to pass to tool

Value

(COVSearchResults) COVsearch tool result object

Examples

```

## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
search_space <- 'COVARIATE(c(CL, V), c(AGE, WT), EXP)'
res <- run_covsearch(model=model, results=results, search_space=search_space)

## End(Not run)

```

run_estmethod

run_estmethod

Description

Run estmethod tool.

Usage

```

run_estmethod(
  algorithm,
  methods = NULL,
  solvers = NULL,
  parameter_uncertainty_methods = NULL,

```

```

    compare_ofv = TRUE,
    results = NULL,
    model = NULL,
    ...
)

```

Arguments

algorithm	(str) The algorithm to use (can be 'exhaustive', 'exhaustive_with_update' or 'exhaustive_only_eval')
methods	(array(str) or str (optional)) List of estimation methods to test. Can be specified as 'all', a vector of estimation methods, or NULL (to not test any estimation method)
solvers	(array(str) or str (optional)) List of solvers to test. Can be specified as 'all', a vector of solvers, or NULL (to not test any solver)
parameter_uncertainty_methods	(array(str) or str (optional)) List of parameter uncertainty methods to test. Can be specified as 'all', a vector of uncertainty methods, or NULL (to not evaluate any uncertainty)
compare_ofv	(logical) Whether to compare the OFV between candidates. Comparison is made by evaluating using IMP
results	(ModelfitResults (optional)) Results for model
model	(Model (optional)) PharmPy mode
...	Arguments to pass to tool

Value

(EstMethodResults) Estmethod tool result object

Examples

```

## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
methods <- c('IMP', 'SAEM')
parameter_uncertainty_methods <- NULL
run_estmethod(
  'reduced', methods=methods, solvers='all',
  parameter_uncertainty_methods=parameter_uncertainty_methods, results=results, model=model
)

## End(Not run)

```

run_iivsearch	<i>run_iivsearch</i>
---------------	----------------------

Description

Run IIVsearch tool. For more details, see :ref:iivsearch.

Usage

```
run_iivsearch(
    model,
    results,
    algorithm = "top_down_exhaustive",
    search_space = "IIV?(@IIV,EXP);COVARIANCE?(IIV,@IIV)",
    as_fullblock = FALSE,
    rank_type = "bic",
    linearize = FALSE,
    cutoff = NULL,
    strictness = "minimization_successful or (rounding_errors and sigdigs>=0.1)",
    correlation_algorithm = NULL,
    E_p = NULL,
    E_q = NULL,
    parameter_uncertainty_method = NULL,
    ...
)
```

Arguments

model	(Model) Pharmpy model
results	(ModelfitResults) Results for model
algorithm	(str) Which algorithm to run when determining number of IIVs.
search_space	(str) Search space to explore
as_fullblock	(logical) Whether to add IIVs as a fullblock
rank_type	(str) Which ranking type should be used. Default is BIC.
linearize	(logical) Whether or not use linearization when running the tool.
cutoff	(numeric (optional)) Cutoff for which value of the ranking function that is considered significant. Default is NULL (all models will be ranked)
strictness	(str) Strictness criteria
correlation_algorithm	(str (optional)) Which algorithm to run for the determining block structure of added IIVs. If NULL, the algorithm is determined based on the 'algorithm' argument
E_p	(numeric or str (optional)) Expected number of predictors for diagonal elements (used for mBIC). Must be set when using mBIC and when the argument 'algorithm' is not 'skip'

E_q (numeric or str (optional)) Expected number of predictors for off-diagonal elements (used for mBIC). Must be set when using mBIC and when the argument correlation_algorithm is not skip or NULL

parameter_uncertainty_method (str (optional)) Parameter uncertainty method. Will be used in ranking models if strictness includes parameter uncertain

... Arguments to pass to tool

Value

(IIVSearchResults) IIVsearch tool result object

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
run_iivsearch(model=model, results=results, algorithm='top_down_exhaustive')

## End(Not run)
```

run_iovsearch	<i>run_iovsearch</i>
---------------	----------------------

Description

Run IOVsearch tool. For more details, see :ref:iovsearch.

Usage

```
run_iovsearch(
  model,
  results,
  column = "OCC",
  list_of_parameters = NULL,
  rank_type = "bic",
  cutoff = NULL,
  distribution = "same-as-iiv",
  strictness = "minimization_successful or (rounding_errors and sigdigs>=0.1)",
  E = NULL,
  parameter_uncertainty_method = NULL,
  ...
)
```

Arguments

model	(Model) Pharmpy model
results	(ModelfitResults) Results for model
column	(str) Name of column in dataset to use as occasion column (default is 'OCC')
list_of_parameters	(array(str or array(str)) (optional)) List of parameters to test IOV on, if none all parameters with IIV will be tested (default)
rank_type	(str) Which ranking type should be used. Default is BIC.
cutoff	(numeric (optional)) Cutoff for which value of the ranking type that is considered significant. Default is NULL (all models will be ranked)
distribution	(str) Which distribution added IOVs should have (default is same-as-iiv)
strictness	(str (optional)) Strictness criteria
E	(numeric or str (optional)) Expected number of predictors (used for mBIC). Must be set when using mBIC
parameter_uncertainty_method	(str (optional)) Parameter uncertainty method. Will be used in ranking models if strictness includes parameter uncertain
...	Arguments to pass to tool

Value

(IOVSearchResults) IOVSearch tool result object

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
run_iovsearch(model=model, results=results, column='OCC')

## End(Not run)
```

run_linearize

run_linearize

Description

Linearize a model

Usage

```
run_linearize(
  model = NULL,
  results = NULL,
  model_name = "linbase",
  description = "",
  ...
)
```

Arguments

model	(Model (optional)) Pharmpy model. Should use FOCEI or other EM-methods for estimation
results	(ModelfitResults (optional)) Results of estimation of model
model_name	(str) New name of linearized model. The default is "linbase".
description	(str) Description of linearized model. The default is ""
...	Arguments to pass to tool

Value

(LinearizeResults) Linearize tool results object.

run_modelfit	<i>run_modelfit</i>
--------------	---------------------

Description

Run modelfit tool.

note:: For most use cases the :func:pharmpy.tools.fit function is a more user friendly option for fitting a model.

Usage

```
run_modelfit(model_or_models = NULL, n = NULL, ...)
```

Arguments

model_or_models	(Model or array(Model) (optional)) A vector of models are one single model object
n	(numeric (optional)) Number of models to fit. This is only used if the tool is going to be combined with other tools
...	Arguments to pass to tool

Value

(ModelfitResults) Modelfit tool result object

Examples

```
## Not run:
model <- load_example_model("pheno")
run_modelfit(model)

## End(Not run)
```

run_modelrank	<i>run_modelrank</i>
---------------	----------------------

Description

Run ModelRank tool.

Usage

```
run_modelrank(
  models,
  results,
  ref_model,
  strictness = "minimization_successful or (rounding_errors and sigdigs >= 0.1)",
  rank_type = "ofv",
  alpha = 0.05,
  search_space = NULL,
  E = NULL,
  parameter_uncertainty_method = NULL,
  exclude_reference_model = FALSE,
  ...
)
```

Arguments

models	(array(Model)) Models to rank
results	(array(ModelfitResults)) Modelfit results to rank on
ref_model	(Model) Model to compare to
strictness	(str) Strictness criteria
rank_type	(str) Which ranking type should be used. Supported types are OFV, LRT, AIC, BIC (mixed, IIV, random), and mBIC (mixed, IIV, random). Default is OFV.
alpha	(numeric (optional)) Cutoff p-value that is considered significant in likelihood ratio test. Default is NULL

search_space	(str or ModelFeatures (optional)) Search space to test. Either as a string or a ModelFeatures object.
E	(numeric or str or list(numeric or str,numeric or str) (optional)) Expected number of predictors (used for mBIC). Must be set when using mBIC. Tuple if mBIC for IIV (both diagonals and off-diagonals)
parameter_uncertainty_method	(str (optional)) Parameter uncertainty method. Will be used in ranking models if strictness includes parameter uncertainty
exclude_reference_model	(logical) Should it be possible to select the reference model? Default is TRUE
...	Arguments to pass to tool

Value

(ModelRankResults) ModelRank tool result object

run_modelsearch	<i>run_modelsearch</i>
-----------------	------------------------

Description

Run Modelsearch tool. For more details, see :ref:modelsearch.

Usage

```
run_modelsearch(
  model,
  results,
  search_space,
  algorithm = "reduced_stepwise",
  iiv_strategy = "absorption_delay",
  rank_type = "bic",
  cutoff = NULL,
  strictness = "minimization_successful or (rounding_errors and sigdigs >= 0.1)",
  E = NULL,
  parameter_uncertainty_method = NULL,
  ...
)
```

Arguments

model	(Model) Pharnpy model
results	(ModelfitResults) Results for model
search_space	(str or ModelFeatures) Search space to test. Either as a string or a ModelFeatures object.

algorithm	(str) Algorithm to use.
iiv_strategy	(str) If/how IIV should be added to candidate models. Default is 'absorption_delay'.
rank_type	(str) Which ranking type should be used. Default is BIC.
cutoff	(numeric (optional)) Cutoff for which value of the ranking function that is considered significant. Default is NULL (all models will be ranked)
strictness	(str) Strictness criteria
E	(numeric or str (optional)) Expected number of predictors (used for mBIC). Must be set when using mBIC
parameter_uncertainty_method	(str (optional)) Parameter uncertainty method. Will be used in ranking models if strictness includes parameter uncertainty
...	Arguments to pass to tool

Value

(ModelSearchResults) Modelsearch tool result object

Examples

```
## Not run:
model <- load_example_model("pheno")
res <- load_example_modelfit_results("pheno")
search_space <- 'ABSORPTION(Z0);PERIPHERALS(1)'
run_modelsearch(model=model, results=res, search_space=search_space, algorithm='exhaustive')

## End(Not run)
```

run_pdsearch

run_pdsearch

Description

Build a PD model

Usage

```
run_pdsearch(
  input,
  type,
  treatment_variable = NULL,
  kpd_driver = "ir",
  algorithm = "stepwise",
  data_strategy = "full",
  results = NULL,
  strictness = "minimization_successful or (rounding_errors and sigdigs>=0.1)",
```

```

    parameter_uncertainty_method = NULL,
    ...
)

```

Arguments

input	(str or Model) A PD/KPD dataset or PD/KPD model
type	(str) Type of PD model to build ('pd' or 'kpd')
treatment_variable	(str (optional)) Name of the variable representing the treatment, e.g. TRT, DOSE or AUC. Do not use if type is 'kpd'
kpd_driver	(str) Driver for KPD model (virtual infusion rate 'ir' or 'amount')
algorithm	(str) Which search algorithm to use. Either 'stepwise' or 'exhaustive_stepwise'
data_strategy	(str) Strategy for using the dataset: 'full', 'partial' or 'fix'
results	(ModelfitResults (optional)) Results to input model
strictness	(str) Strictness criteria
parameter_uncertainty_method	(str (optional)) Parameter uncertainty method. Will be used in ranking models if strictness includes parameter uncertain
...	Arguments to pass to tool

Value

(PDSearchResults) PDSearch tool results object.

run_qa

run_qa

Description

Run QA tool.

Usage

```
run_qa(model = NULL, results = NULL, linearize = FALSE, skip = NULL, ...)
```

Arguments

model	(Model (optional)) Pharnmpy model
results	(ModelfitResults (optional)) Results of model
linearize	(logical) Whether or not to use linearization when running the tool
skip	(array(str) (optional)) A vector of sections to ski
...	Arguments to pass to tool

Value

(QAResults) QA tool result object

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
run_qa(model=model, results=results, linearize=FALSE, skip=c('fullblock'))

## End(Not run)
```

run_retries

run_retries

Description

Run retries tool.

Usage

```
run_retries(
  model = NULL,
  results = NULL,
  number_of_candidates = 5,
  fraction = 0.1,
  use_initial_estimates = FALSE,
  strictness = "minimization_successful or (rounding_errors and sigdigs >= 0.1)",
  scale = "UCP",
  prefix_name = "",
  parameter_uncertainty_method = NULL,
  ...
)
```

Arguments

model	(Model (optional)) Model object to run retries on. The default is NULL.
results	(ModelfitResults (optional)) Connected ModelfitResults object. The default is NULL.
number_of_candidates	(numeric) Number of retry candidates to run. The default is 5.
fraction	(numeric) Determines allowed increase/decrease from initial parameter estimate. Default is 0.1 (10%)
use_initial_estimates	(logical) Use initial parameter estimates instead of final estimates of input model when creating candidate models.

strictness	(str) Strictness criteria. The default is "minimization_successful or (rounding_errors and sigdigs >= 0.1)".
scale	(str (optional)) Which scale to update the initial values on. Either normal scale or UCP scale.
prefix_name	(str (optional)) Prefix the candidate model names with given string.
parameter_uncertainty_method	(str (optional)) Parameter uncertainty method. Will be used in ranking models if strictness includes parameter uncertain
...	Arguments to pass to tool

Value

(RetriesResults) Retries tool results object.

run_ruvsearch	<i>run_ruvsearch</i>
---------------	----------------------

Description

Run the ruvsearch tool. For more details, see :ref:ruvsearch.

Usage

```
run_ruvsearch(
  model,
  results,
  groups = 4,
  p_value = 0.001,
  skip = NULL,
  max_iter = 3,
  dv = NULL,
  strictness = "minimization_successful or (rounding_errors and sigdigs>=0.1)",
  parameter_uncertainty_method = NULL,
  ...
)
```

Arguments

model	(Model) Pharnpy model
results	(ModelfitResults) Results of model
groups	(numeric) The number of bins to use for the time varying models
p_value	(numeric) The p-value to use for the likelihood ratio test
skip	(array(str) (optional)) A vector of models to not attempt.
max_iter	(numeric) Number of iterations to run (1, 2, or 3). For models with BLQ only one iteration is supported.

```

dv          (numeric (optional)) Which DV to assess the error model for.
strictness  (str) Strictness criteria
parameter_uncertainty_method
            (str (optional)) Parameter uncertainty method. Will be used in ranking models
            if strictness includes parameter uncertain
...        Arguments to pass to tool

```

Value

(RUVSearchResults) Ruvsearch tool result object

Examples

```

## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
run_ruvsearch(model=model, results=results)

## End(Not run)

```

run_simulation	<i>run_simulation</i>
----------------	-----------------------

Description

Run the simulation tool.

Usage

```
run_simulation(model = NULL, ...)
```

Arguments

```

model      (Model (optional)) Pharmpy mode
...        Arguments to pass to tool

```

Value

(SimulationResult) SimulationResults object

Examples

```

## Not run:
model <- load_example_model("pheno")
model <- set_simulation(model, n=10)
run_simulations(model)

## End(Not run)

```

run_structsearch	<i>run_structsearch</i>
------------------	-------------------------

Description

Run the structsearch tool. For more details, see :ref:structsearch.

Usage

```
run_structsearch(
    model,
    results,
    type,
    search_space = NULL,
    b_init = NULL,
    emax_init = NULL,
    ec50_init = NULL,
    met_init = NULL,
    extra_model = NULL,
    rank_type = "bic",
    cutoff = NULL,
    strictness = "minimization_successful or (rounding_errors and sigdigs >= 0.1)",
    extra_model_results = NULL,
    dv_types = NULL,
    parameter_uncertainty_method = NULL,
    ...
)
```

Arguments

model	(Model) Pharnpy start model
results	(ModelfitResults) Results for the start model
type	(str) Type of model. Currently only 'drug_metabolite', 'pkpd' and 'tmdd'
search_space	(str or ModelFeatures (optional)) Search space to test
b_init	(numeric (optional)) Initial estimate for the baseline for pkpd models.
emax_init	(numeric (optional)) Initial estimate for E_MAX (for pkpd models only).
ec50_init	(numeric (optional)) Initial estimate for EC_50 (for pkpd models only).
met_init	(numeric (optional)) Initial estimate for MET (for pkpd models only).
extra_model	(Model (optional)) Optional extra Pharnpy model to use in TMDD structsearch
rank_type	(str) Results for the extra model
cutoff	(numeric (optional)) Which ranking type should be used. Default is BIC.
strictness	(str (optional)) Cutoff for which value of the ranking function that is considered significant. Default is NULL (all models will be ranked)

```

extra_model_results      (ModelfitResults (optional)) Strictness criteria
dv_types                  (list(str=numeric) (optional)) Dictionary of DV types for TMDD models with
                          multiple DVs
parameter_uncertainty_method (str (optional)) Parameter uncertainty method. Will be used in ranking models
                          if strictness includes parameter uncertain
...                       Arguments to pass to tool

```

Value

(StructSearchResult) structsearch tool result object

Examples

```

## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
run_structsearch(model=model, results=results, model_type='pkpd')

## End(Not run)

```

run_tool

run_tool

Description

Run tool workflow

note:: This is a general function that can run any tool. There is also one function for each specific tool. Please refer to the documentation of these for more specific information.

Usage

```
run_tool(tool_name, ...)
```

Arguments

```

tool_name      (str) Name of tool to run
...           Arguments to pass to tool

```

Value

(Results) Results object for tool

Examples

```
## Not run:
model <- load_example_model("pheno")
res <- run_tool("ruvsearch", model)

## End(Not run)
```

run_vpc

run_vpc

Description

Run VPC

Usage

```
run_vpc(
  model,
  results = NULL,
  samples = 20,
  stratify = NULL,
  frem = FALSE,
  ...
)
```

Arguments

model	(Model) PharmPy model
results	(ModelfitResults (optional)) Results for model
samples	(numeric) Number of samples
stratify	(str (optional)) Column to stratify on
frem	(logical) Should we run the special vpc procedure a FREM model
...	Arguments to pass to tool

Value

(VPCResults) VPC results object

Examples

```
## Not run:
model <- load_example_model("pheno")
res <- load_example_modelfit_results("pheno")
run_vpc(model, res, samples=200)

## End(Not run)
```

```
sample_individual_estimates  
    sample_individual_estimates
```

Description

Sample individual estimates given their covariance.

Usage

```
sample_individual_estimates(  
  model,  
  individual_estimates,  
  individual_estimates_covariance,  
  parameters = NULL,  
  samples_per_id = 100,  
  seed = 1234  
)
```

Arguments

model	(Model) PharmPy model
individual_estimates	(data.frame) Individual estimates to use
individual_estimates_covariance	(data.frame) Uncertainty covariance of the individual estimates
parameters	(array(str) (optional)) A vector of a subset of individual parameters to sample. Default is NULL, which means all.
samples_per_id	(numeric) Number of samples per individual
seed	(numeric) Random number generator or seed

Value

(data.frame) Pool of samples in a DataFrame

See Also

`sample_parameters_from_covariance_matrix` : Sample parameter vectors using the uncertainty covariance matrix

`sample_parameters_uniformly` : Sample parameter vectors using uniform distribution

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
rng <- create_rng(23)
ie <- results$individual_estimates
iec <- results$individual_estimates_covariance
sample_individual_estimates(model, ie, iec, samples_per_id=2, seed=rng)

## End(Not run)
```

```
sample_parameters_from_covariance_matrix
  sample_parameters_from_covariance_matrix
```

Description

Sample parameter vectors using the covariance matrix

If parameters is not provided all estimated parameters will be used

Usage

```
sample_parameters_from_covariance_matrix(
  model,
  parameter_estimates,
  covariance_matrix,
  force_posdef_samples = NULL,
  force_posdef_covmatrix = FALSE,
  n = 1,
  seed = 1234
)
```

Arguments

model	(Model) PharmPy model
parameter_estimates	(list(str=numeric)) Parameter estimates to use as means in sampling
covariance_matrix	(data.frame) Parameter uncertainty covariance matrix
force_posdef_samples	(numeric (optional)) Set to how many iterations to do before forcing all samples to be positive definite. NULL is default and means never and 0 means always
force_posdef_covmatrix	(logical) Set to TRUE to force the input covariance matrix to be positive definite
n	(numeric) Number of samples
seed	(numeric) Random number generator

Value

(data.frame) A dataframe with one sample per row

See Also

sample_parameters_uniformly : Sample parameter vectors using uniform distribution

sample_individual_estimates : Sample individual estimates given their covariance

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_model_fit_results("pheno")
rng <- create_rng(23)
cov <- results$covariance_matrix
pe <- results$parameter_estimates
sample_parameters_from_covariance_matrix(model, pe, cov, n=3, seed=rng)

## End(Not run)
```

sample_parameters_uniformly
sample_parameters_uniformly

Description

Sample parameter vectors using uniform sampling

Each parameter value will be randomly sampled from a uniform distribution with the bounds being estimate \pm estimate * fraction.

Usage

```
sample_parameters_uniformly(  
  model,  
  parameter_estimates,  
  fraction = 0.1,  
  force_posdef_samples = NULL,  
  n = 1,  
  seed = 1234,  
  scale = "normal"  
)
```

Arguments

model	(Model) Pharmpy model
parameter_estimates	(list(str=numeric)) Parameter estimates for parameters to use
fraction	(numeric) Fraction of estimate value to use for distribution bounds
force_posdef_samples	(numeric (optional)) Number of samples to reject before forcing variability parameters to give positive definite covariance matrices.
n	(numeric) Number of samples
seed	(numeric) Random number generator or seed
scale	(str) Scale to perform sampling on. Valid options are 'normal' and 'UCP'

Value

(data.frame) samples

See Also

sample_parameters_from_covariance_matrix : Sample parameter vectors using the uncertainty covariance matrix

sample_individual_estimates : Sample individual estimates given their covariance

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_model_fit_results("pheno")
rng <- create_rng(23)
pe <- results$parameter_estimates
sample_parameters_uniformly(model, pe, n=3, seed=rng)

## End(Not run)
```

```
set_additive_error_model
      set_additive_error_model
```

Description

Set an additive error model. Initial estimate for new sigma is (equation could not be rendered, see API doc on website)

The error function being applied depends on the data transformation. The table displays some examples.

```

+-----+-----+ | Data transformation | Additive error |
+=====+=====+ | (equation could not be rendered, see API doc on website) |
-----+ | (equation could not be rendered, see API doc on website) |
-----+

```

Usage

```
set_additive_error_model(model, dv = NULL, data_trans = NULL, series_terms = 2)
```

Arguments

model	(Model) Pharmpy model
dv	(Expr or str or numeric (optional)) Name or DVID of dependent variable. NULL for the default (first or only)
data_trans	(numeric or str or Expr (optional)) A data transformation expression or NULL (default) to use the transformation specified by the model. Series expansion will be used for approximation.
series_terms	(numeric) Number of terms to use for the series expansion approximation for data transformation.

Value

(Model) Updated Pharmpy model

See Also

set_proportional_error_model : Proportional error model

set_combined_error_model : Combined error model

Examples

```

## Not run:
model <- load_example_model("pheno")
model$statements$find_assignment("Y")
model <- set_additive_error_model(model)
model$statements$find_assignment("Y")
model <- load_example_model("pheno")
model$statements$find_assignment("Y")
model <- set_additive_error_model(model, data_trans="log(Y)")
model$statements$find_assignment("Y")

## End(Not run)

```


Usage

```
set_combined_error_model(model, dv = NULL, data_trans = NULL)
```

Arguments

model	(Model) Pharmpy model
dv	(Expr or str or numeric (optional)) Name or DVID of dependent variable. NULL for the default (first or only)
data_trans	(numeric or str or Expr (optional)) A data transformation expression or NULL (default) to use the transformation specified by the model.

Value

(Model) Updated Pharmpy model

See Also

set_additive_error_model : Additive error model
 set_proportional_error_model: Proportional error model

Examples

```
## Not run:
model <- remove_error_model(load_example_model("pheno"))
model <- set_combined_error_model(model)
model$statements$find_assignment("Y")
model <- remove_error_model(load_example_model("pheno"))
model <- set_combined_error_model(model, data_trans="log(Y)")
model$statements$find_assignment("Y")

## End(Not run)
```

set_covariates	<i>set_covariates</i>
----------------	-----------------------

Description

Set columns in the dataset to be covariates in the datainfo

Usage

```
set_covariates(model, covariates)
```

Arguments

model	(Model) Pharmpy model
covariates	(array(str)) Column names

Value

(Model) Updated Pharmpy model

set_dataset	<i>set_dataset</i>
-------------	--------------------

Description

Load the dataset given datainfo

Usage

```
set_dataset(model, path_or_df, datatype = NULL)
```

Arguments

model	(Model) Pharmpy model
path_or_df	(str or data.frame) Dataset path or dataframe
datatype	(str (optional)) Type of dataset (optional)

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- unload_dataset(model)  
dataset_path <- model$datainfo$path  
model$dataset is NULL  
model <- set_dataset(model, dataset_path, datatype='nonmem')  
model$dataset  
  
## End(Not run)
```

set_description *set_description*

Description

Set description of model object

Usage

```
set_description(model, new_description)
```

Arguments

```
model                    (Model) Pharmpy model
new_description           (str) New description of model
```

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model$description
model <- set_description(model, "PHENOBARB run 2")
model$description

## End(Not run)
```

set_direct_effect *set_direct_effect*

Description

Add an effect to a model.

Effects are by default using concentration, but any user specified variable in the model can be used. Implemented PD models are:

- Linear:

(equation could not be rendered, see API doc on website)

- Emax:

(equation could not be rendered, see API doc on website)

- Step effect:

(equation could not be rendered, see API doc on website)

- Sigmoidal:

(equation could not be rendered, see API doc on website)

- Log-linear:

(equation could not be rendered, see API doc on website)

(equation could not be rendered, see API doc on website)

Usage

```
set_direct_effect(model, expr, variable = NULL)
```

Arguments

model	(Model) Pharmpy model
expr	(str) Name of PD effect function.
variable	(str (optional)) Name of variable to use (if NULL concentration will be used)

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_direct_effect(model, "linear")
model$statements$find_assignment("E")

## End(Not run)
```

```
set_dtbs_error_model  set_dtbs_error_model
```

Description

Dynamic transform both sides

Usage

```
set_dtbs_error_model(model, fix_to_log = FALSE)
```

Arguments

model	(Model) Pharmpy model
fix_to_log	(logical) Set to TRUE to fix lambda and zeta to 0, i.e. emulating log-transformed data

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- set_dtbs_error_model(model)  
  
## End(Not run)
```

set_dvid

set_dvid

Description

Set a column to act as DVID. Replace DVID if one is already set.

Usage

```
set_dvid(model, name)
```

Arguments

model	(Model) Pharmpy model
name	(str) Name of DVID column

Value

(Model) Updated Pharmpy model

set_estimation_step *set_estimation_step*

Description

Set estimation step

Sets estimation step for a model. Methods currently supported are: FO, FOCE, ITS, LAPLACE, IMPMAP, IMP, SAEM, BAYES

Usage

```
set_estimation_step(model, method, idx = 0, ...)
```

Arguments

model	(Model) Pharmpy model
method	(str) estimation method to change to
idx	(numeric) index of estimation step, default is 0 (first estimation step)
...	Arguments to pass to EstimationStep (such as interaction, evaluation)

Value

(Model) Updated Pharmpy model

See Also

add_estimation_step
remove_estimation_step
append_estimation_step_options
add_parameter_uncertainty_step
remove_parameter_uncertainty_step
set_evaluation_step

Examples

```
## Not run:  
model <- load_example_model("pheno")  
opts <- list('NITER'=1000, 'ISAMPLE'=100)  
model <- set_estimation_step(model, 'IMP', evaluation=TRUE, tool_options=opts)  
model$execution_steps[1]  
  
## End(Not run)
```

set_evaluation_step *set_evaluation_step*

Description

Set evaluation step

Change the final or the estimation step with a specific index to do evaluation.

Usage

```
set_evaluation_step(model, idx = -1)
```

Arguments

model (Model) Pharmpy model

idx (numeric) Index of estimation step, default is -1 (last estimation step)

Value

(Model) Update Pharmpy model

See Also

set_estimation_step

add_estimation_step

remove_estimation_step

append_estimation_step_options

add_parameter_uncertainty_step

remove_parameter_uncertainty_step

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- set_evaluation_step(model)  
model$execution_steps[1]  
  
## End(Not run)
```

```
set_first_order_absorption  
    set_first_order_absorption
```

Description

Set or change to first order absorption rate.

Initial estimate for absorption rate is set to the previous rate if available, otherwise it is set to the time of first observation/2.

If multiple doses is set to the affected compartment, currently only iv+oral doses (one of each) is supported

Usage

```
set_first_order_absorption(model)
```

Arguments

model (Model) Pharmpy model

Value

(Model) Updated Pharmpy model

See Also

set_instantaneous_absorption

set_zero_order_absorption

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- set_first_order_absorption(model)  
model$statements$ode_system  
  
## End(Not run)
```

```
set_first_order_elimination  
    set_first_order_elimination
```

Description

Sets elimination to first order

Usage

```
set_first_order_elimination(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) Updated PharmPy model

See Also

set_zero_order_elimination
set_michaelis_menten_elimination

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- set_first_order_elimination(model)  
model$statements$ode_system  
  
## End(Not run)
```

```
set_iiv_on_ruv            set_iiv_on_ruv
```

Description

Multiplies epsilons with exponential (new) etas.
Initial variance for new etas is 0.09.

Usage

```
set_iiv_on_ruv(
  model,
  dv = NULL,
  list_of_eps = NULL,
  same_eta = TRUE,
  eta_names = NULL
)
```

Arguments

model	(Model) Pharmpy model
dv	(Expr or numeric (optional)) Name/names of epsilons to multiply with exponential etas. If NULL, all epsilons will be chosen. NULL is default.
list_of_eps	(array(str) or str (optional)) Boolean of whether all RUVs from input should use the same new ETA or if one ETA should be created for each RUV. TRUE is default.
same_eta	(logical) Custom names of new etas. Must be equal to the number epsilons or 1 if same eta.
eta_names	(array(str) or str (optional)) Name or DVID of dependent variable. NULL for the default (first or only)

Value

(Model) Updated Pharmpy model

See Also

set_power_on_ruv

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_iiv_on_ruv(model)
model$statements$find_assignment("Y")

## End(Not run)
```

set_initial_condition *set_initial_condition*

Description

Set an initial condition for the ode system

If the initial condition is already set it will be updated. If the initial condition is set to zero at time zero it will be removed (since the default is 0).

Usage

```
set_initial_condition(model, compartment, expression, time = 0)
```

Arguments

model	(Model) Pharmpy model
compartment	(str) Name of the compartment
expression	(numeric or str or Expr) The expression of the initial condition
time	(numeric or str or Expr) Time point. Default 0

Value

(model) Updated Pharmpy model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- set_initial_condition(model, "CENTRAL", 10)  
get_initial_conditions(model)  
  
## End(Not run)
```

set_initial_estimates *set_initial_estimates*

Description

Update initial parameter estimate for a model

Updates initial estimates of population parameters for a model. If the new initial estimates are out of bounds or NaN this function will raise.

Usage

```
set_initial_estimates(  
  model,  
  inits,  
  move_est_close_to_bounds = FALSE,  
  strict = TRUE  
)
```

Arguments

model	(Model) Pharmpy model
inits	(list(str=numeric)) Initial parameter estimates to update
move_est_close_to_bounds	(logical) Move estimates that are close to bounds. If correlation >0.99 the correlation will be set to 0.9, if variance is <0.001 the variance will be set to 0.01.
strict	(logical) Whether all parameters in input need to exist in the model. Default is TRUE Setting strict to FALSE will also disregard any initial estimate being NaN and keep the original value for these parameters.

Value

(Model) Updated Pharmpy model

See Also

fix_parameters_to : Fixing and setting parameter initial estimates in the same function
 unfix_paramaters_to : Unfixing parameters and setting a new initial estimate in the same

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_model_fit_results("pheno")
model$parameters$inits
model <- set_initial_estimates(model, results$parameter_estimates)
model$parameters$inits
model <- load_example_model("pheno")
model <- set_initial_estimates(model, list('POP_CL'=2.0))
model$parameters['POP_CL']

## End(Not run)
```

```
set_instantaneous_absorption
      set_instantaneous_absorption
```

Description

Set or change to instantaneous absorption rate.
 Currently lagtime together with instantaneous absorption is not supported.

Usage

```
set_instantaneous_absorption(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) Updated PharmPy model

See Also

set_zero_order_absorption

set_first_order_absorption

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_instantaneous_absorption(model)
model$statements$ode_system

## End(Not run)
```

set_lloq_data	<i>set_lloq_data</i>
---------------	----------------------

Description

Set a dv value for lloq data records

Usage

```
set_lloq_data(model, value, lloq = NULL, blq = NULL)
```

Arguments

model (Model) PharmPy model

value (str or numeric or Expr) The new dv value

lloq (numeric or str (optional)) Value or column name for lower limit of quantification.

blq (str (optional)) Column name for below limit of quantification indicator.

Value

(Model) Updated PharmPy model

See Also

remove_loq_data

transform_blq

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_lloq_data(model, 0, lloq=10)

## End(Not run)
```

set_lower_bounds *set_lower_bounds*

Description

Set parameter lower bounds

Usage

```
set_lower_bounds(model, bounds, strict = TRUE)
```

Arguments

model	(Model) PharmPy model
bounds	(list(str=numeric)) A list of parameter bounds for parameters to change
strict	(logical) Whether all parameters in input need to exist in the model. Default is TRUE

Value

(Model) Updated PharmPy model

See Also

set_upper_bounds : Set parameter upper bounds
unconstrain_parameters : Remove all constraints of parameters

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_lower_bounds(model, {'POP_CL': -10})
model$parameters['POP_CL']

## End(Not run)
```

```
set_michaelis_menten_elimination  
    set_michaelis_menten_elimination
```

Description

Sets elimination to Michaelis-Menten.

Note that the parametrization is not the usual, but is instead using a CLMM parameter.

Initial estimate for CLMM is set to CL and KM is set to (equation could not be rendered, see API doc on website)

Usage

```
set_michaelis_menten_elimination(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) Updated PharmPy model

See Also

set_first_order_elimination

set_zero_order_elimination

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- set_michaelis_menten_elimination(model)  
model$statements$ode_system  
  
## End(Not run)
```

```
set_mixed_mm_fo_elimination  
    set_mixed_mm_fo_elimination
```

Description

Sets elimination to mixed Michaelis-Menten and first order.

Initial estimate for CLMM is set to CL/2 and KM is set to (equation could not be rendered, see API doc on website)

Usage

```
set_mixed_mm_fo_elimination(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) Updated PharmPy model

See Also

set_first_order_elimination

set_zero_order_elimination

set_michaelis_menten_elimination

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- set_mixed_mm_fo_elimination(model)  
model$statements$ode_system  
  
## End(Not run)
```

set_name	<i>set_name</i>
----------	-----------------

Description

Set name of model object

Usage

```
set_name(model, new_name)
```

Arguments

model	(Model) Pharmpy model
new_name	(str) New name of model

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model$name
model <- set_name(model, "run2")
model$name

## End(Not run)
```

set_n_transit_compartments	<i>set_n_transit_compartments</i>
----------------------------	-----------------------------------

Description

Set the n-transit compartments model

This is the absorption delay model where the number of transit compartments is a parameter to be estimated (1) (2). Initial estimate for absorption rate is set the previous rate if available, otherwise it is set to the time of first observation/2. Initial estimate for the number of transit compartments is set to 2.

Currently only handles a single oral route of administration. Assumes complete absorption between doses

(1) Savic, R.M., Jonker, D.M., Kerbusch, T. et al. Implementation of a transit compartment model for describing drug absorption in pharmacokinetic studies. *J Pharmacokinet Pharmacodyn* 34, 711–726 (2007). <https://doi.org/10.1007/s10928-007-9066-0>

(2) Shen, J., Boeckmann, A. & Vick, A. Implementation of dose superimposition to introduce multiple doses for a mathematical absorption model (transit compartment model). *J Pharmacokinet Pharmacodyn* 39, 251–262 (2012). <https://doi.org/10.1007/s10928-012-9247-3>

Usage

```
set_n_transit_compartments(model, keep_depot = TRUE)
```

Arguments

model	(Model) PharmPy model
keep_depot	(logical) FALSE to convert depot compartment into a transit compartment

Value

(Model) Updated PharmPy model

See Also

`add_lag_time`
`set_transit_compartments`

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- set_n_transit_compartments(model)  
  
## End(Not run)
```

set_ode_solver	<i>set_ode_solver</i>
----------------	-----------------------

Description

Sets ODE solver to use for model

The recognized solvers are CVODES, DGEAR, DVERK, IDA, LSODA and LSODI. To see the corresponding NONMEM ADVANS see `:ref:ADVANS`.

Usage

```
set_ode_solver(model, solver)
```

Arguments

model (Model) PharmPy model
 solver (str) Solver to use or NULL for no preference

Value

(Model) Updated PharmPy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_ode_solver(model, 'LSODA')

## End(Not run)
```

set_peripheral_compartments
set_peripheral_compartments

Description

Sets the number of peripheral compartments for central compartment to a specified number.
 If name is set, the peripheral compartment will be added to the compartment with the specified name instead.

Usage

```
set_peripheral_compartments(model, n, name = NULL)
```

Arguments

model (Model) PharmPy model
 n (numeric) Number of transit compartments
 name (str (optional)) Name of compartment to add peripheral to.

Value

(Model) Updated PharmPy model

See Also

add_peripheral_compartment
 remove_peripheral_compartment

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_peripheral_compartments(model, 2)
model$statements$ode_system

## End(Not run)
```

```
set_placebo_model      set_placebo_model
```

Description

Add a placebo or disease progression effect to a model.

warning:: This function is under development.

- linear

(equation could not be rendered, see API doc on website)

- exp_decrease

(equation could not be rendered, see API doc on website)

- exp_increase

(equation could not be rendered, see API doc on website)

- tmax

(equation could not be rendered, see API doc on website)

(equation could not be rendered, see API doc on website)

Usage

```
set_placebo_model(model, expr, operator = "*")
```

Arguments

model	(Model) PharmPy model
expr	(str) Name of placebo/disease progression effect function.
operator	(str) Operator to use for combining the baseline with the placebo/disease progression

Value

(Model) Updated PharmPy model

Examples

```
## Not run:
model <- create_basic_pd_model()
model <- set_placebo_model(model, "linear")
model$statements$find_assignment("PDP")

## End(Not run)
```

```
set_power_on_ruv      set_power_on_ruv
```

Description

Applies a power effect to provided epsilons. If a dependent variable is provided, then only said epsilons affecting said variable will be changed.

Initial estimates for new thetas are 1 if the error model is proportional, otherwise they are 0.1.

NOTE : If no DVs or epsilons are specified, all epsilons with the same name will be connected to the same theta. Running the function per DV will give each epsilon a specific theta.

Usage

```
set_power_on_ruv(
  model,
  list_of_eps = NULL,
  dv = NULL,
  lower_limit = 0.01,
  ipred = NULL,
  zero_protection = FALSE
)
```

Arguments

model	(Model) Pharmpy model
list_of_eps	(str or array (optional)) Name/names of epsilons to apply power effect. If NULL, all epsilons will be used. NULL is default.
dv	(Expr or numeric (optional)) Name or DVID of dependent variable. NULL will change the epsilon on all occurrences regardless of affected dependent variable.
lower_limit	(numeric (optional)) Lower limit of power (theta). NULL for no limit.
ipred	(str or Expr (optional)) Symbol to use as IPRED. Default is to autodetect expression for IPRED.
zero_protection	(logical) Set to TRUE to add code protecting from IPRED=0

Value

(Model) Updated Pharmpy model

See Also

set_iiv_on_ruv

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_power_on_ruv(model)
model$statements$find_assignment("Y")

## End(Not run)
```

set_property

set_property

Description

Specify a property of a column

See `:py:attr:pharmpy.DataInfo.properties` for documentation on data properties.

Usage

```
set_property(model_or_datainfo, column, property, value)
```

Arguments

model_or_datainfo	(Model or DataInfo) Model object or DataInfo object
column	(str) Name of a column. If the column contains multiple variables, e.g. DV with multiple DVs, the ID can be specified with a colon. For example "DV:1" will mean the DV column only when DVID is 1.
property	(str) Name of the property to set
value	(any) Value of the property to set

Value

(Model | DataInfo) An updated Model or DataInfo object

See Also

set_unit - Set unit of a data variable

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_property(model, "APGR", "categories", c(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10))

## End(Not run)
```

```
set_proportional_error_model
      set_proportional_error_model
```

Description

Set a proportional error model. Initial estimate for new sigma is 0.09.

The error function being applied depends on the data transformation.

```
+-----+-----+ | Data transformation | Proportional error
| +=====+ | (equation could not be rendered, see API doc on website) + |
-----+ | (equation could not be rendered, see API doc on website) + |
-----+
```

Usage

```
set_proportional_error_model(
  model,
  dv = NULL,
  data_trans = NULL,
  zero_protection = TRUE
)
```

Arguments

model	(Model) PharmPy model
dv	(Expr or str or numeric (optional)) Name or DVID of dependent variable. NULL for the default (first or only)
data_trans	(numeric or str or Expr (optional)) A data transformation expression or NULL (default) to use the transformation specified by the model.
zero_protection	(logical) Set to TRUE to add code protecting from IPRED=0

Value

(Model) Updated PharmPy model

See Also

set_additive_error_model : Additive error model
set_combined_error_model : Combined error model

Examples

```
## Not run:
model <- remove_error_model(load_example_model("pheno"))
model <- set_proportional_error_model(model)
model$statements$after_odes
model <- remove_error_model(load_example_model("pheno"))
model <- set_proportional_error_model(
  model,
  data_trans="log(Y)"
model$statements$after_odes

## End(Not run)
```

set_reference_values *set_reference_values*

Description

Set reference values for selected columns

All values for each selected column will be replaced. For dose columns only the values for dosing events will be replaced.

Usage

```
set_reference_values(model, refs)
```

Arguments

model (Model) PharmPy model
refs (list(str=numeric)) Pairs of column names and reference values

Value

(Model) Updated PharmPy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_reference_values(model, list('WGT'=0.5, 'AMT'=4.0))
model$dataset

## End(Not run)
```

```
set_seq_zo_fo_absorption  
    set_seq_zo_fo_absorption
```

Description

Set or change to sequential zero order first order absorption rate.

Initial estimate for absorption rate is set the previous rate if available, otherwise it is set to the time of first observation/2.

Currently lagtime together with sequential zero order first order absorption is not supported.

Usage

```
set_seq_zo_fo_absorption(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) Updated PharmPy model

See Also

set_instantaneous_absorption

set_zero_order_absorption

set_first_order_absorption

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- set_seq_zo_fo_absorption(model)  
model$statements$ode_system  
  
## End(Not run)
```

set_simulation	<i>set_simulation</i>
----------------	-----------------------

Description

Change model into simulation model

Usage

```
set_simulation(model, n = 1, seed = 1234)
```

Arguments

model	(Model) Pharmpy model
n	(numeric) Number of replicates
seed	(numeric) Random seed for the simulation

Value

(Model) Update Pharmpy model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- set_simulation(model, n=10, seed=1234)  
steps <- model$execution_steps  
steps[1]  
  
## End(Not run)
```

set_time_varying_error_model	<i>set_time_varying_error_model</i>
------------------------------	-------------------------------------

Description

Set a time varying error model per time cutoff

Usage

```
set_time_varying_error_model(model, cutoff, idv = "TIME", dv = NULL)
```

Arguments

model	(Model) Pharmpy model
cutoff	(numeric) A cutoff value for idv column
idv	(str) Time or time after dose, default is Time
dv	(Expr or str or numeric (optional)) Name or DVID of dependent variable. NULL for the default (first or only)

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_time_varying_error_model(model, cutoff=1.0)
model$statements$find_assignment("Y")

## End(Not run)
```

set_tmdd

set_tmdd

Description

Sets target mediated drug disposition

Implemented target mediated drug disposition (TMDD) models are:

- Full model
- Irreversible binding approximation (IB)
- Constant total receptor approximation (CR)
- Irreversible binding and constant total receptor approximation (CR+IB)
- Quasi steady-state approximation (QSS)
- Wagner
- Michaelis-Menten approximation (MMAPP)

Usage

```
set_tmdd(model, type, dv_types = NULL)
```

Arguments

model	(Model) Pharmpy model
type	(str) Type of TMDD model
dv_types	(list(str=numeric) (optional)) Dictionary of DV types for TMDD models with multiple DVs (e.g. <code>dv_types = list('drug' = 1, 'target' = 2)</code>). Default is NULL which means that all observations are treated as drug observations. For <code>dv = 1</code> the only allowed keys are 'drug' and 'drug_tot'. If no DV for drug is specified then (free) drug will have <code>dv = 1</code> .

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_tmdd(model, "full")

## End(Not run)
```

```
set_transit_compartments
      set_transit_compartments
```

Description

Set the number of transit compartments of model.

Initial estimate for the absorption delay is set the previous delay if available, otherwise it is set to the time of first observation/2.

Usage

```
set_transit_compartments(model, n, keep_depot = TRUE)
```

Arguments

model	(Model) Pharmpy model
n	(numeric) Number of transit compartments
keep_depot	(logical) FALSE to convert depot compartment into a transit compartment

Value

(Model) Updated Pharmpy model

See Also

add_lag_time
 set_n_transit_compartments

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_transit_compartments(model, 3)
model$statements$ode_system

## End(Not run)
```

set_unit	<i>set_unit</i>
----------	-----------------

Description

Specify the unit of a column

Note that no conversion of units will happen if the unit was already set.

Usage

```
set_unit(model_or_datainfo, column, unit)
```

Arguments

model_or_datainfo	(Model or DataInfo) Model object or DataInfo object
column	(str) Name of a column. If the column contains multiple variables, e.g. DV with multiple DVs, the ID can be specified with a colon. For example "DV:1" will mean the DV column only when DVID is 1.
unit	(str) The unit

Value

(Model | DataInfo) An updated Model or DataInfo object

See Also

convert_unit - Convert between units for a variable

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_unit(model, "WGT", "kg")

## End(Not run)
```

set_upper_bounds	<i>set_upper_bounds</i>
------------------	-------------------------

Description

Set parameter upper bounds

Usage

```
set_upper_bounds(model, bounds, strict = TRUE)
```

Arguments

model	(Model) PharmPy model
bounds	(list(str=numeric)) A list of parameter bounds for parameters to change
strict	(logical) Whether all parameters in input need to exist in the model. Default is TRUE

Value

(Model) Updated PharmPy model

See Also

set_lower_bounds : Set parameter lower bounds
unconstrain_parameters : Remove all constraints of parameters

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- set_upper_bounds(model, list('POP_CL'=10))  
model$parameters['POP_CL']  
  
## End(Not run)
```

```
set_weibull_absorption
      set_weibull_absorption
```

Description

Set or change to Weibull type absorption

The Weibull absorption has an absorption rate varying with time (or rather time after dose). (equation could not be rendered, see API doc on website)

Initial parameter estimates will be set differently depending on whether the original model has MAT and/or MDT.

```
=====
MAT MDT (equation could not be rendered, see API doc on website) === === =====
===== yes yes 1.5 (equation
could not be rendered, see API doc on website) yes no 1.0 MAT no yes 1.0 MDT no no 1.5 Same as if
having MAT, but use min observation time * 2 =====
```

If multiple doses are fed into the affected compartment, currently only iv+oral doses (one of each) is supported.

Weibull absorption cannot be used together with lag time or transit compartments.

Assumes that absorption of one dose is complete when the next dose is given.

Usage

```
set_weibull_absorption(model)
```

Arguments

```
model          (Model) PharmPy model
```

Value

```
(Model) Updated PharmPy model
```

See Also

```
set_zero_order_absorption
```

```
set_first_order_absorption
```

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_weibull_absorption(model)
model$statements$code_system
```

```
## End(Not run)
```

```
set_weighted_error_model  
    set_weighted_error_model
```

Description

Encode error model with one epsilon and W as weight

Usage

```
set_weighted_error_model(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) Updated PharmPy model

See Also

use_thetas_for_error_stdev : Use thetas to estimate error

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- set_weighted_error_model(model)  
  
## End(Not run)
```

```
set_zero_order_absorption  
    set_zero_order_absorption
```

Description

Set or change to zero order absorption rate.

Initial estimate for absorption rate is set the previous rate if available, otherwise it is set to the time of first observation/2.

Usage

```
set_zero_order_absorption(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) Updated PharmPy model

See Also

set_instantaneous_absorption

set_first_order_absorption

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- set_zero_order_absorption(model)  
model$statements$ode_system  
  
## End(Not run)
```

set_zero_order_elimination

set_zero_order_elimination

Description

Sets elimination to zero order.

Initial estimate for KM is set to 1% of smallest observation.

Usage

```
set_zero_order_elimination(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) Updated PharmPy model

See Also

set_first_order_elimination

set_michaelis_menten_elimination

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_zero_order_elimination(model)
model$statements$ode_system

## End(Not run)
```

set_zero_order_input *set_zero_order_input*

Description

Set a zero order input for the ode system
If the zero order input is already set it will be updated.

Usage

```
set_zero_order_input(model, compartment, expression)
```

Arguments

model	(Model) PharmPy model
compartment	(str) Name of the compartment
expression	(numeric or str or Expr) The expression of the zero order input

Value

(model) Updated PharmPy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_zero_order_input(model, "CENTRAL", 10)
get_zero_order_inputs(model)

## End(Not run)
```

simplify_expression *simplify_expression*

Description

Simplify expression given constraints in model

Usage

```
simplify_expression(model, expr)
```

Arguments

model	(Model) PharmPy model
expr	(str or numeric or Expr) Expression to simplify

Value

(Expression) Simplified expression

Examples

```
## Not run:  
model <- load_example_model("pheno")  
simplify_expression(model, "Abs(POP_CL)")  
  
## End(Not run)
```

solve_ode_system *solve_ode_system*

Description

Replace ODE system with analytical solution if possible

Warnings This function can currently only handle the most simple of ODE systems.

Usage

```
solve_ode_system(model)
```

Arguments

model	(Model) PharmPy model object
-------	------------------------------

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model$statements$ode_system
model <- solve_ode_system(model)

## End(Not run)
```

split_joint_distribution

split_joint_distribution

Description

Splits etas following a joint distribution into separate distributions.

Usage

```
split_joint_distribution(model, rvs = NULL)
```

Arguments

model	(Model) Pharmpy model
rvs	(array(str) or str (optional)) Name/names of etas or individual parameters to separate. If NULL, all etas that are IIVs and non-fixed will become single. NULL is default.

Value

(Model) Updated Pharmpy model

See Also

create_joint_distribution : combine etas into a join distribution

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- create_joint_distribution(model, c('ETA_CL', 'ETA_VC'))
model$random_variables$etas
model <- split_joint_distribution(model, c('ETA_CL', 'ETA_VC'))
model$random_variables$etas

## End(Not run)
```

```
summarize_modelfit_results
      summarize_modelfit_results
```

Description

Summarize results of model runs

Summarize different results after fitting a model, includes runtime, ofv, and parameter estimates (with errors). If `include_all_execution_steps` is `FALSE`, only the last estimation step will be included (note that in that case, the `minimization_successful` value will be referring to the last estimation step, if last step is evaluation it will go backwards until it finds an estimation step that wasn't an evaluation).

Usage

```
summarize_modelfit_results(context, include_all_execution_steps = FALSE)
```

Arguments

`context` (Context) Context in which models were run
`include_all_execution_steps` (logical) Whether to include all estimation steps, default is `FALSE`

Value

(data.frame) A DataFrame of modelfit results with model name and estimation step as index.

```
transform_blq      transform_blq
```

Description

Transform for BLQ data

Transform a given model, methods available are m1, m3, m4, m5, m6 and m7 (1). The blq information can come from the dataset, the lloq option or a combination. Both LLOQ and BLQ columns are supported. The table below explains which columns are used for the various cases:

	LLOQ column	BLQ column	Used as indicator	Used as level	Note	lloq option
Available	NA	NA	DV < lloq	lloq		
NA	NA	Available	NA	DV < LLOQ	LLOQ	
nothing	Only for M1 and M7					NA
NA	NA	NA	NA	NA	No BLQ handling	
NA	Available	Available	BLQ	LLOQ		

DV column not used	Available	NA	Available	BLQ	lloq	Column overridden
Available	Available	Available	DV < lloq	lloq	Columns overridden	

BLQ observations are defined as shown in the table above. If both a BLQ and an LLOQ column exist in the dataset and no lloq is specified then all dv values in rows with BLQ = 1 are counted as BLQ observations. If instead an lloq value is specified then all rows with dv values below the lloq value are counted as BLQ observations. If no lloq is specified and no BLQ column exists in the dataset then all rows with dv values below the value specified in the LLOQ column are counted as BLQ observations.

M1 method: All BLQ observations are discarded. This may affect the size of the dataset. M3 method: Including the probability that the BLQ observations are below the LLOQ as part of the maximum likelihood estimation. For more details see :ref:(1)<ref_article>. This method modifies the Y statement of the model (see examples below). M4 method: Including the probability that the BLQ observations are below the LLOQ and positive as part of the maximum likelihood estimation. For more details see :ref:(1)<ref_article>. This method modifies the Y statement of the model (see examples below). M5 method: All BLQ observations are replaced by level/2, where level = lloq if lloq is specified. Else level = value specified in LLOQ column (see table above). This method may change entries in the dataset. M6 method: Every BLQ observation in a consecutive series of BLQ observations is discarded except for the first one. The remaining BLQ observations are replaced by level/2, where level = lloq if lloq is specified. Else level = value specified in LLOQ column (see table above). This method may change entries in the dataset as well as the size of the dataset. M7 method: All BLQ observations are replaced by 0. This method may change entries in the dataset.

Current limitations of the m3 and m4 method:

- Does not support covariance between epsilons
- Supports additive, proportional, combined, and power error model

ref_article:

(1) Beal SL. Ways to fit a PK model with some data below the quantification limit. J Pharmacokinetic Pharmacodyn. 2001 Oct;28(5):481-504. doi: 10.1023/a:1012299115260. Erratum in: J Pharmacokinetic Pharmacodyn 2002 Jun;29(3):309. PMID: 11768292.

Usage

```
transform_blq(model, method = "m4", lloq = NULL)
```

Arguments

model	(Model) PharmPy model
method	(str) Which BLQ method to use
lloq	(numeric (optional)) LLOQ limit to use, if NULL PharmPy will use the BLQ/LLOQ column in the dataset

Value

(Model) Updated Pharmpy model

See Also

remove_loq_data

set_loq_data

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- transform_blq(model, method='m4', lloq=0.1)
model$statements$find_assignment("Y")

## End(Not run)
```

transform_etas_boxcox *transform_etas_boxcox*

Description

Applies a boxcox transformation to selected etas

Initial estimate for lambda is 0.1 with bounds (-3, 3).

Usage

```
transform_etas_boxcox(model, list_of_etas = NULL)
```

Arguments

model (Model) Pharmpy model

list_of_etas (array(str) or str (optional)) Name/names of etas to transform. If NULL, all etas will be transformed (default).

Value

(Model) Updated Pharmpy model

See Also

transform_etas_tdist

transform_etas_john_draper

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- transform_etas_boxcox(model, c("ETA_CL"))
model$statements$before_odes$full_expression("CL")

## End(Not run)
```

```
transform_etas_john_draper
      transform_etas_john_draper
```

Description

Applies a John Draper transformation to selected etas

See (1) for more information.

Initial estimate for lambda is 0.1 with bounds (-3, 3).

(1) John, J., Draper, N. (1980). An Alternative Family of Transformations. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 29(2), 190-197. doi:10.2307/2986305

Usage

```
transform_etas_john_draper(model, list_of_etas = NULL)
```

Arguments

model	(Model) Pharmpy model
list_of_etas	(array(str) or str (optional)) Name/names of etas to transform. If NULL, all etas will be transformed (default).

Value

(Model) Updated Pharmpy model

See Also

```
transform_etas_boxcox
transform_etas_tdist
```

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- transform_etas_john_draper(model, c("ETA_CL"))
model$statements$before_odes$full_expression("CL")

## End(Not run)
```

transform_etas_tdist *transform_etas_tdist*

Description

Applies a t-distribution transformation to selected etas

Initial estimate for degrees of freedom is 80 with bounds (3, 100).

Usage

```
transform_etas_tdist(model, list_of_etas = NULL)
```

Arguments

model (Model) Pharmpy model

list_of_etas (array(str) or str (optional)) Name/names of etas to transform. If NULL, all etas will be transformed (default).

Value

(Model) Updated Pharmpy model

See Also

transform_etas_boxcox

transform_etas_john_draper

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- transform_etas_tdist(model, c("ETA_CL"))
model$statements$before_odes$full_expression("CL")

## End(Not run)
```

```
translate_nmtran_time translate_nmtran_time
```

Description

Translate NM-TRAN TIME and DATE column into one TIME column

If dataset of model have special NM-TRAN TIME and DATE columns these will be translated into one single time column with time in hours.

Warnings Use this function with caution. For example reset events are currently not taken into account.

Usage

```
translate_nmtran_time(model)
```

Arguments

model (Model) Pharmpy model

Value

(Model) Updated Pharmpy model

```
unconstrain_parameters  
unconstrain_parameters
```

Description

Remove all constraints from parameters

Usage

```
unconstrain_parameters(model, parameter_names, strict = TRUE)
```

Arguments

model (Model) Pharmpy model

parameter_names

(array(str)) Remove all constraints for the listed parameters

strict

(logical) Whether all parameters in input need to exist in the model. Default is TRUE

Value

(Model) Updated Pharmpy model

See Also

set_lower_bounds : Set parameter lower bounds
 set_upper_bounds : Set parameter upper bounds
 unfix_parameters : Unfix parameters

Examples

```
## Not run:
model <- load_example_model("pheno")
model$parameters['POP_CL']
model <- unconstrain_parameters(model, c('POP_CL'))
model$parameters['POP_CL']

## End(Not run)
```

undrop_columns	<i>undrop_columns</i>
----------------	-----------------------

Description

Undrop columns of model

Usage

```
undrop_columns(model, column_names)
```

Arguments

model (Model) Pharmpy model
 column_names (array(str) or str) List of column names or one column name to undrop

Value

(Model) Updated Pharmpy model

See Also

drop_dropped_columns : Drop all columns marked as drop
 drop_columns : Drop or mark columns as dropped

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- drop_columns(model, c('WGT', 'APGR'), mark=TRUE)
model <- undrop_columns(model, 'WGT')

## End(Not run)
```

unfix_parameters	<i>unfix_parameters</i>
------------------	-------------------------

Description

Unfix parameters

Unfix all listed parameters

Usage

```
unfix_parameters(model, parameter_names, strict = TRUE)
```

Arguments

model (Model) Pharmpy model

parameter_names

(array(str) or str) one parameter name or a vector of parameter names

strict

(logical) Whether all parameters in input need to exist in the model. Default is TRUE

Value

(Model) Updated Pharmpy model

See Also

unfix_parameters_to : Unfixing parameters and setting a new initial estimate in the same function

fix_parameters : Fix parameters

fix_or_unfix_parameters : Fix or unfix parameters (given boolean)

fix_parameters_to : Fixing and setting parameter initial estimates in the same function

unconstrain_parameters : Remove all constraints of parameters

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- fix_parameters(model, c('POP_CL', 'POP_VC'))
model$parameters$fix
model <- unfix_parameters(model, 'POP_CL')
model$parameters$fix

## End(Not run)
```

unfix_parameters_to *unfix_parameters_to*

Description

Unfix parameters to
Unfix all listed parameters to specified value/values

Usage

```
unfix_parameters_to(model, inits, strict = TRUE)
```

Arguments

model	(Model) Pharmpy model
inits	(list(str=numeric)) Inits for all parameters to unfix and change init
strict	(logical) Whether all parameters in input need to exist in the model. Default is TRUE

Value

(Model) Updated Pharmpy model

See Also

fix_parameters : Fix parameters
fix_or_unfix_parameters : Fix or unfix parameters (given boolean)
unfix_paramaters : Unfixing parameters
fix_paramaters_to : Fixing parameters and setting a new initial estimate in the same function

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- fix_parameters(model, c('POP_CL', 'POP_VC'))  
model$parameters$fix  
model <- unfix_parameters_to(model, list('POP_CL'=0.5))  
model$parameters$fix  
model$parameters['POP_CL']  
  
## End(Not run)
```

unload_dataset	<i>unload_dataset</i>
----------------	-----------------------

Description

Unload the dataset from a model

Usage

```
unload_dataset(model)
```

Arguments

model (Model) PharmPy model

Value

(Model) Updated PharmPy model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- unload_dataset(model)  
model$dataset is NULL  
  
## End(Not run)
```

update_initial_individual_estimates	<i>update_initial_individual_estimates</i>
-------------------------------------	--

Description

Update initial individual estimates for a model

Updates initial individual estimates for a model.

Usage

```
update_initial_individual_estimates(model, individual_estimates, force = TRUE)
```

Arguments

model (Model) Pharmpy model
 individual_estimates (array) Individual estimates to use
 force (logical) Set to FALSE to only update if the model had initial individual estimates before

Value

(Model) Updated Pharmpy model

Examples

```

## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
ie <- results$individual_estimates
model <- update_initial_individual_estimates(model, ie)

## End(Not run)

```

```

use_thetas_for_error_stdev
      use_thetas_for_error_stdev

```

Description

Use thetas to estimate standard deviation of error

Usage

```
use_thetas_for_error_stdev(model)
```

Arguments

model (Model) Pharmpy model

Value

(Model) Updated Pharmpy model

See Also

set_weighted_error_model : Encode error model with one epsilon and weight

`write_csv`*write_csv*

Description

Write dataset to a csv file and updates the datainfo path

Usage

```
write_csv(model, path = NULL, force = FALSE)
```

Arguments

<code>model</code>	(Model) Pharmpy model
<code>path</code>	(str (optional)) Destination path. Default is to use original path with .csv suffix.
<code>force</code>	(logical) Overwrite file with same path. Default is FALSE.

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
model <- write_csv(model, path="newdataset$csv")  
  
## End(Not run)
```

`write_dataset`*write_dataset*

Description

Write dataset to file and updates the datainfo path

Currently supports csv files.

Usage

```
write_dataset(model, path = NULL, force = FALSE, type = "csv")
```

Arguments

model	(Model) Pharmpy model
path	(str (optional)) Destination path. Default is to use original path with .csv suffix.
force	(logical) Overwrite file with same path. Default is FALSE.
type	(str) Can only be csv

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- write_dataset(model, path="newdataset$csv")

## End(Not run)
```

write_model

write_model

Description

Write model code to file

An updated Pharmpy model is returned. This will have a new name based on the filename and it might have updates to the model code (e.g. for \$DATA in NONMEM models).

Usage

```
write_model(model, path = "", force = TRUE)
```

Arguments

model	(Model) Pharmpy model
path	(str) Destination path
force	(logical) Force overwrite, default is TRUE

Value

(Model) Updated Pharmpy model

Examples

```
## Not run:  
model <- load_example_model("pheno")  
write_model(model)  
  
## End(Not run)
```

write_results	<i>write_results</i>
---------------	----------------------

Description

Write results object to json (or csv) file

Note that the csv-file cannot be read into a results object again.

Usage

```
write_results(results, path, compression = FALSE, csv = FALSE)
```

Arguments

results	(Results) PharmPy results object
path	(str) Path to results file
compression	(logical) TRUE to compress the file. Not applicable to csv file
csv	(logical) Save as csv file

Index

add_admid, 8
add_allometry, 9
add_bioavailability, 10
add_cmt, 11
add_covariate_effect, 12
add_derivative, 14
add_effect_compartment, 15
add_estimation_step, 16
add_iiv, 18
add_indirect_effect, 20
add_individual_parameter, 21
add_iov, 22
add_lag_time, 23
add_metabolite, 24
add_output_variables, 24
add_parameter_uncertainty_step, 25
add_pd_iiv, 26
add_peripheral_compartment, 27
add_pk_iiv, 28
add_population_parameter, 29
add_predictions, 30
add_residuals, 31
add_time_after_dose, 32
add_time_of_last_dose, 32
append_estimation_step_options, 33

bin_observations, 35
binarize_dataset, 34
broadcast_log, 36
bump_model_number, 36

calculate_aic, 37
calculate_bic, 37
calculate_corr_from_cov, 38
calculate_corr_from_prec, 39
calculate_cov_from_corrse, 40
calculate_cov_from_prec, 41
calculate_epsilon_gradient_expression, 42
calculate_eta_gradient_expression, 43

calculate_eta_shrinkage, 44
calculate_individual_parameter_statistics, 45
calculate_individual_shrinkage, 46
calculate_parameters_from_ucp, 47
calculate_pk_parameters_statistics, 48
calculate_prec_from_corrse, 49
calculate_prec_from_cov, 50
calculate_se_from_cov, 51
calculate_se_from_prec, 52
calculate_summary_statistic, 53
calculate_ucp_scale, 54
check_dataset, 54
check_high_correlations, 55
check_parameters_near_bounds, 56
check_pharmpy, 57
check_setup, 57
cholesky_decompose, 57
cleanup_model, 58
convert_model, 59
convert_unit, 60
create_basic_kpd_model, 60
create_basic_pd_model, 61
create_basic_pk_model, 62
create_config_template, 63
create_joint_distribution, 63
create_report, 64
create_rng, 64
create_symbol, 65

deidentify_data, 66
display_odes, 66
drop_columns, 67
drop_dropped_columns, 68

evaluate_epsilon_gradient, 68
evaluate_eta_gradient, 69
evaluate_expression, 70
evaluate_individual_prediction, 71
evaluate_population_prediction, 72

- evaluate_weighted_residuals, 73
- expand_additional_doses, 74
- export_model_files, 74

- filter_dataset, 75
- find_clearance_parameters, 76
- find_volume_parameters, 76
- fit, 77
- fix_or_unfix_parameters, 78
- fix_parameters, 79
- fix_parameters_to, 80

- get_admid, 81
- get_baselines, 81
- get_bioavailability, 82
- get_central_volume_and_clearance, 82
- get_cmt, 83
- get_column_name, 83
- get_concentration_parameters_from_data, 84
- get_config_path, 85
- get_covariate_baselines, 85
- get_covariate_effects, 86
- get_doseid, 86
- get_doses, 87
- get_dv_symbol, 88
- get_evid, 88
- get_ids, 89
- get_individual_parameters, 89
- get_individual_prediction_expression, 90
- get_initial_conditions, 91
- get_lag_times, 92
- get_mdv, 92
- get_model_code, 93
- get_model_covariates, 93
- get_mu_connected_to_parameter, 94
- get_nested_model, 95
- get_number_of_individuals, 96
- get_number_of_observations, 97
- get_number_of_observations_per_individual, 98
- get_number_of_peripheral_compartments, 99
- get_number_of_transit_compartments, 99
- get_observation_expression, 101
- get_observations, 100
- get_omegas, 101
- get_parameter_rv, 102

- get_pd_parameters, 103
- get_pk_parameters, 104
- get_population_prediction_expression, 105
- get_rv_parameters, 105
- get_sigmas, 106
- get_thetas, 107
- get_unit_of, 108
- get_zero_order_inputs, 108
- greekify_model, 109

- has_additive_error_model, 110
- has_combined_error_model, 111
- has_covariate_effect, 112
- has_first_order_absorption, 112
- has_first_order_elimination, 113
- has_instantaneous_absorption, 114
- has_linear_odes, 114
- has_linear_odes_with_real_eigenvalues, 115
- has_michaelis_menten_elimination, 116
- has_mixed_mm_fo_elimination, 116
- has_mu_reference, 117
- has_odes, 118
- has_presystemic_metabolite, 118
- has_proportional_error_model, 119
- has_random_effect, 120
- has_seq_zo_fo_absorption, 121
- has_weibull_absorption, 121
- has_weighted_error_model, 122
- has_zero_order_absorption, 122
- has_zero_order_elimination, 123

- infer_datatypes, 124
- insert_ebes_into_dataset, 124
- install_pharmpy, 125
- install_pharmpy_devel, 126
- is_binary, 126
- is_linearized, 127
- is_real, 128
- is_simulation_model, 128
- is_strictness_fulfilled, 129

- list_models, 130
- list_time_varying_covariates, 130
- load_dataset, 131
- load_example_model, 132
- load_example_modelfit_results, 132
- make_declarative, 133

- map_eta_parameters, 134
- mu_reference_model, 134

- omit_data, 135
- open_context, 136

- plot_abs_cwres_vs_ipred, 136
- plot_cwres_vs_idv, 137
- plot_dv_vs_ipred, 138
- plot_dv_vs_pred, 139
- plot_eta_distributions, 139
- plot_individual_predictions, 140
- plot_iofv_vs_iofv, 141
- plot_transformed_eta_distributions, 142
- plot_vpc, 142
- predict_influential_individuals, 144
- predict_influential_outliers, 144
- predict_outliers, 145
- print_fit_summary, 146
- print_log, 147
- print_model_code, 147
- print_model_symbols, 148
- print_pharmpy_version, 148

- read_dataset_from_datainfo, 149
- read_model, 149
- read_model_from_string, 150
- read_modelfit_results, 150
- read_results, 151
- remove_bioavailability, 152
- remove_covariate_effect, 153
- remove_derivative, 153
- remove_error_model, 154
- remove_estimation_step, 155
- remove_iiv, 156
- remove_iov, 157
- remove_lag_time, 158
- remove_loq_data, 158
- remove_parameter_uncertainty_step, 160
- remove_peripheral_compartment, 161
- remove_predictions, 162
- remove_residuals, 163
- remove_unused_columns, 164
- remove_unused_parameters_and_rvs, 164
- rename_symbols, 165
- replace_fixed_thetas, 165
- replace_non_random_rvs, 166
- resample_data, 166

- reset_index, 167
- reset_indices_results, 168
- retrieve_model, 168
- retrieve_modelfit_results, 169
- retrieve_models, 169
- run_allometry, 170
- run_amd, 171
- run_bootstrap, 173
- run_covsearch, 174
- run_estmethod, 176
- run_iivsearch, 178
- run_iovsearch, 179
- run_linearize, 180
- run_modelfit, 181
- run_modelrank, 182
- run_modelsearch, 183
- run_pdsearch, 184
- run_qa, 185
- run_retries, 186
- run_ruvsearch, 187
- run_simulation, 188
- run_structsearch, 189
- run_tool, 190
- run_vpc, 191

- sample_individual_estimates, 192
- sample_parameters_from_covariance_matrix, 193
- sample_parameters_uniformly, 194
- set_additive_error_model, 195
- set_baseline_effect, 197
- set_combined_error_model, 197
- set_covariates, 198
- set_dataset, 199
- set_description, 200
- set_direct_effect, 200
- set_dtbs_error_model, 201
- set_dvid, 202
- set_estimation_step, 203
- set_evaluation_step, 204
- set_first_order_absorption, 205
- set_first_order_elimination, 206
- set_iiv_on_ruv, 206
- set_initial_condition, 207
- set_initial_estimates, 208
- set_instantaneous_absorption, 209
- set_lloq_data, 210
- set_lower_bounds, 211
- set_michaelis_menten_elimination, 212

set_mixed_mm_fo_elimination, 213
set_n_transit_compartments, 214
set_name, 214
set_ode_solver, 215
set_peripheral_compartments, 216
set_placebo_model, 217
set_power_on_ruv, 218
set_property, 219
set_proportional_error_model, 220
set_reference_values, 221
set_seq_zo_fo_absorption, 222
set_simulation, 223
set_time_varying_error_model, 223
set_tmdd, 224
set_transit_compartments, 225
set_unit, 226
set_upper_bounds, 227
set_weibull_absorption, 228
set_weighted_error_model, 229
set_zero_order_absorption, 229
set_zero_order_elimination, 230
set_zero_order_input, 231
simplify_expression, 232
solve_ode_system, 232
split_joint_distribution, 233
summarize_modelfit_results, 234

transform_bfq, 234
transform_etas_boxcox, 236
transform_etas_john_draper, 237
transform_etas_tdist, 238
translate_nmtran_time, 239

unconstrain_parameters, 239
undrop_columns, 240
unfix_parameters, 241
unfix_parameters_to, 242
unload_dataset, 243
update_initial_individual_estimates,
 243
use_thetas_for_error_stdev, 244

write_csv, 245
write_dataset, 245
write_model, 246
write_results, 247