Package ‘pharmr’

June 17, 2024

Encoding UTF-8

Version 1.0.1

Date 2024-06-12

Title Interface to the 'Pharmpy' 'Pharmacometrics' Library

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

Depends R (>= 3.6.0), altair (>= 4.0.0)

SystemRequirements Python (>= 3.10.0)

Imports reticulate (>= 1.19), 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.

Config/reticulate list(packages = list(list(package = ``altair"), list(package = ``pharmpy-core")))

URL https://github.com/pharmpy/pharmr

BugReports https://github.com/pharmpy/pharmr/issues

License LGPL (>= 3)

RoxygenNote 7.3.1

Author Rikard Nordgren [aut, cre, cph],
Stella Belin [aut, cph],
Mats O. Karlsson [sad],
Andrew C. Hooker [sad],
Xiaomei Chen [sad],
Sebastian Ueckert [sad] (<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 2024-06-17 10:00:02 UTC

Contents

add_admid . . . 7
add_allometry . . . 8
add_bioavailability . . . 9
add_cmt . . . 10
add_covariate_effect . . . 11
add_derivative . . . 13
add_effect_compartment . . . 14
add_estimation_step . . . 15
add_iiv . . . 16
add_indirect_effect . . . 18
add_individual_parameter . . . 19
add_iov . . . 20
add_lag_time . . . 21
add_metabolite . . . 22
add_parameter_uncertainty_step . . . 22
add_pd_iiv . . . 23
add_peripheral_compartment . . . 24
add_pk_iiv . . . 25
add_population_parameter . . . 26
add_predictions . . . 27
add_residuals . . . 28
add_time_after_dose . . . 29
append_estimation_step_options . . . 30
bin_observations . . . 31
bump_model_number . . . 32
calculate_aic . . . 32
calculate_bic . . . 33
calculate_corr_from_cov . . . 34
calculate_corr_from_prec . . . 35
calculate_cov_from_corrse . . . 36
calculate_cov_from_prec . . . 37
calculate_epsilon_gradient_expression . . . 38
calculate_eta_gradient_expression . . . 38
calculate_eta_shrinkage . . . 39
calculate_individual_parameter_statistics . . . 40
calculate_individual_shrinkage . . . 41
calculate_parameters_from_ucp . . . 42
calculate_pk_parameters_statistics . . . 43
calculate_prec_from_corrse . . . 44
calculate_prec_from_cov . . . 45
calculate_se_from_cov . . . 46
calculate_se_from_prec . . . 47
calculate_ucp_scale . . . 48
check_dataset . . . 48
check_high_correlations . . . 49
check_parameters_near_bounds . . . 50

check_pharmpy . . . 51
cleanup_model . . . 51
convert_model . . . 52
create_basic_pk_model . . . 53
create_config_template . . . 54
create_joint_distribution . . . 54
create_report . . . 55
create_rng . . . 55
create_symbol . . . 56
deidentify_data . . . 57
display_odes . . . 57
drop_columns . . . 58
drop_dropped_columns . . . 59
evaluate_epsilon_gradient . . . 59
evaluate_eta_gradient . . . 60
evaluate_expression . . . 61
evaluate_individual_prediction . . . 62
evaluate_population_prediction . . . 63
evaluate_weighted_residuals . . . 64
expand_additional_doses . . . 65
filter_dataset . . . 65
find_clearance_parameters . . . 66
find_volume_parameters . . . 67
fit . . . 67
fix_or_unfix_parameters . . . 68
fix_parameters . . . 69
fix_parameters_to . . . 70
get_admid . . . 71
get_baselines . . . 71
get_bioavailability . . . 72
get_central_volume_and_clearance . . . 72
get_cmt . . . 73
get_concentration_parameters_from_data . . . 73
get_config_path . . . 74
get_covariate_baselines . . . 74
get_covariate_effects . . . 75
get_doseid . . . 76
get_doses . . . 76
get_dv_symbol . . . 77
get_evid . . . 78
get_ids . . . 78
get_individual_parameters . . . 79
get_individual_prediction_expression . . . 80
get_initial_conditions . . . 80
get_lag_times . . . 81
get_mdv . . . 82
get_model_code . . . 82
get_model_covariates . . . 83

get_number_of_individuals . . . 83
get_number_of_observations . . . 84
get_number_of_observations_per_individual . . . 85
get_number_of_peripheral_compartments . . . 86
get_number_of_transit_compartments . . . 86
get_observations . . . 87
get_observation_expression . . . 87
get_omegas . . . 88
get_parameter_rv . . . 89
get_pd_parameters . . . 90
get_pk_parameters . . . 90
get_population_prediction_expression . . . 91
get_rv_parameters . . . 92
get_sigmas . . . 93
get_thetas . . . 93
get_unit_of . . . 94
get_zero_order_inputs . . . 95
greekify_model . . . 95
has_additive_error_model . . . 96
has_combined_error_model . . . 97
has_covariate_effect . . . 98
has_first_order_absorption . . . 98
has_first_order_elimination . . . 99
has_instantaneous_absorption . . . 100
has_linear_odes . . . 100
has_linear_odes_with_real_eigenvalues . . . 101
has_michaelis_menten_elimination . . . 102
has_mixed_mm_fo_elimination . . . 102
has_odes . . . 103
has_presystemic_metabolite . . . 104
has_proportional_error_model . . . 104
has_random_effect . . . 105
has_seq_zo_fo_absorption . . . 106
has_weighted_error_model . . . 107
has_zero_order_absorption . . . 107
has_zero_order_elimination . . . 108
install_pharmpy . . . 109
install_pharmpy_devel . . . 109
is_linearized . . . 110
is_real . . . 110
is_strictness_fulfilled . . . 111
list_time_varying_covariates . . . 112
load_dataset . . . 112
load_example_model . . . 113
load_example_modelfit_results . . . 114
make_declarative . . . 114
mu_reference_model . . . 115
omit_data . . . 116

plot_abs_cwres_vs_ipred . . . 116
plot_cwres_vs_idv . . . 117
plot_dv_vs_ipred . . . 118
plot_dv_vs_pred . . . 119
plot_eta_distributions . . . 119
plot_individual_predictions . . . 120
plot_iofv_vs_iofv . . . 121
plot_transformed_eta_distributions . . . 122
plot_vpc . . . 122
predict_influential_individuals . . . 123
predict_influential_outliers . . . 124
predict_outliers . . . 125
print_fit_summary . . . 126
print_model_code . . . 126
print_model_symbols . . . 127
print_pharmpy_version . . . 127
read_dataset_from_datainfo . . . 128
read_model . . . 128
read_modelfit_results . . . 129
read_model_from_string . . . 129
read_results . . . 130
remove_bioavailability . . . 131
remove_covariate_effect . . . 131
remove_derivative . . . 132
remove_error_model . . . 133
remove_estimation_step . . . 133
remove_iiv . . . 134
remove_iov . . . 135
remove_lag_time . . . 136
remove_loq_data . . . 137
remove_parameter_uncertainty_step . . . 138
remove_peripheral_compartment . . . 139
remove_predictions . . . 140
remove_residuals . . . 141
remove_unused_parameters_and_rvs . . . 142
rename_symbols . . . 142
replace_non_random_rvs . . . 143
resample_data . . . 143
reset_index . . . 144
reset_indices_results . . . 145
retrieve_models . . . 145
run_allometry . . . 146
run_amd . . . 147
run_bootstrap . . . 149
run_covsearch . . . 150
run_estmethod . . . 151
run_iivsearch . . . 152
run_iovsearch . . . 154

run_linearize . . . 155
run_modelfit . . . 155
run_modelsearch . . . 156
run_retries . . . 157
run_ruvsearch . . . 158
run_simulation . . . 159
run_structsearch . . . 160
run_tool . . . 161
sample_individual_estimates . . . 162
sample_parameters_from_covariance_matrix . . . 163
sample_parameters_uniformly . . . 164
set_additive_error_model . . . 166
set_baseline_effect . . . 167
set_combined_error_model . . . 168
set_covariates . . . 169
set_dataset . . . 169
set_direct_effect . . . 170
set_dtbs_error_model . . . 171
set_dvid . . . 172
set_estimation_step . . . 172
set_evaluation_step . . . 173
set_first_order_absorption . . . 174
set_first_order_elimination . . . 175
set_iiv_on_ruv . . . 176
set_initial_condition . . . 177
set_initial_estimates . . . 177
set_instantaneous_absorption . . . 178
set_lloq_data . . . 179
set_lower_bounds . . . 180
set_michaelis_menten_elimination . . . 181
set_mixed_mm_fo_elimination . . . 182
set_name . . . 183
set_ode_solver . . . 183
set_peripheral_compartments . . . 184
set_power_on_ruv . . . 185
set_proportional_error_model . . . 186
set_reference_values . . . 187
set_seq_zo_fo_absorption . . . 188
set_simulation . . . 189
set_time_varying_error_model . . . 190
set_tmdd . . . 190
set_transit_compartments . . . 191
set_upper_bounds . . . 192
set_weighted_error_model . . . 193
set_zero_order_absorption . . . 194
set_zero_order_elimination . . . 194
set_zero_order_input . . . 195
simplify_expression . . . 196

solve_ode_system 197
split_joint_distribution 197
summarize_modelfit_results 198
transform_blq 199
transform_etas_boxcox 200
transform_etas_john_draper 201
transform_etas_tdist 202
translate_nmtran_time 203
unconstrain_parameters 203
undrop_columns 204
unfix_parameters 205
unfix_parameters_to 206
unload_dataset 207
update_initial_individual_estimates 207
use_thetas_for_error_stdev 208
write_csv 209
write_model 209
write_results 210

Index **211**

`add_admid` *add_admid*

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) Pharmpy model

See Also

get_admid : Get or create an admid column

get_cmt : Get or create a cmt column

`add_allometry` *add_allometry*

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) Pharmpy 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) Pharmpy model object

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) Pharmpy model object

See Also

remove_bioavailability

Examples

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

## End(Not run)
```

---

add_cmt *add_cmt*

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) 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 <= 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 to add covariate effect to.
`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) Pharmpy model object

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 calculcated 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 derivates 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 varibles 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 modeas.
`with_respect_to`	(array(array(str) or str) or str (optional)) Parameter name(s) to use as independent variables. Default is NULL.

Value

(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) Pharmpy model object

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, ...)
```

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)
`...`	Arguments to pass to EstimationStep (such as interaction, evaluation)

Value

(Model) Pharmpy model object

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 to add new IIVs to.
`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) Pharmpy model object

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)
```

Arguments

`model`	(Model) Pharmpy model
`expr`	(str) Production (TRUE) (default) or degradation (FALSE)
`prod`	(logical) Name of PD effect function.

Value

(Model) Pharmpy model object

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)
```

Arguments

`model`	(Model) Pharmpy model
`name`	(str) Name of individual/pk parameter

Value

(Model) Pharmpy model object

Examples

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

## End(Not run)
```

add_iov *add_iov*

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 to add new IOVs to.
`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) Pharmpy model object

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) Pharmpy model object

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 wether or not to add metabolite as a presystemetic fixed drug.

Value

(Model) Pharmpy model object

Examples

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

## 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) Pharmpy model object

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 to add new IIVs to.
`initial_estimate`	(numeric) Value of initial estimate of parameter. Default is 0.09

Value

(Model) Pharmpy model object

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 QPn / 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 QPn / VPn where VPn is the volumne 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) Name of compartment to add peripheral to.

Value

(Model) Pharmpy model object

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 to add new IIVs to.
`initial_estimate`	(numeric) Value of initial estimate of parameter. Default is 0.09

Value

(Model) Pharmpy model object

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) Pharmpy model object

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) Pharmpy model object

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 *add_residuals*

Description

Add predictions and/or residuals

Add residuals to estimation step.

Added redidual 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) Pharmpy model object

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) Pharmpy model object

Examples

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

## End(Not run)
```

`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)
```

Arguments

`model`	(Model) Pharmpy model
`tool_options`	(list(str=any)) any additional tool specific options
`idx`	(numeric) index of estimation step (starting from 0)

Value

(Model) Pharmpy model object

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)
```

`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)
```

bump_model_number *bump_model_number*

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

model	(Model) Pharmpy model object
path	(str (optional)) Default is to not look for files.

Value

(Model) Pharmpy model object

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 *calculate_aic*

Description

Calculate AIC

AIC = -2LL + 2*n_estimated_parameters

Usage

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

Arguments

`model`	(Model) Pharmpy model object
`likelihood`	(numeric) -2LL

Value

(numeric) AIC of model fit

`calculate_bic` *calculate_bic*

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 object
`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 object

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 object

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_modelfit_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 = NULL
)
```

Arguments

`model`	(Model) A previously estimated model
`expr_or_exprs`	(array(BooleanExpr) or array(Expr) or array(str) or BooleanExpr or Expr or str) Parameter estimates
`parameter_estimates`	(array) Parameter uncertainty covariance matrix
`covariance_matrix`	(data.frame (optional)) 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.
`seed`	(numeric (optional)) 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: ieta_shr = (var(eta) / 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_modelfit_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 <- {'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 = NULL
)
```

Arguments

model	(Model) A previously estimated model
parameter_estimates	(array) Parameter estimates
covariance_matrix	(data.frame (optional)) Parameter uncertainty covariance matrix
seed	(numeric (optional)) 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_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) Model for which to calculate an ucp scale

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 object
`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 object
`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_modelfit_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 object
`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` *Checks version of Pharmpy/pharmr*

Description

Checks whether Pharmpy and pharmr has the same version

Usage

```
check_pharmpy(pharmpy_version)
```

Arguments

`pharmpy_version`	(str) version number as string

`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

Usage

```
cleanup_model(model)
```

Arguments

`model`	(Model) Pharmpy model object

Value

(Model) Reference to the same model

Note

When creating NONMEM code from the cleaned model Pharmpy might need toadd 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 *convert_model*

Description

Convert model to other format

Note that the operation is not done inplace.

Usage

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

Arguments

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

Value

(Model) New model object with new underlying model format

Examples

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

## 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 object

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) Pharmpy model
`rvs`	(array(str) (optional)) Sequence of etas or names of etas to combine. 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) Pharmpy model object

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` *create_report*

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` *create_rng*

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 = NULL)
```

Arguments

seed	(numeric (optional)) 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)
```

---

create_symbol *create_symbol*

Description

Create a new unique variable symbol given a model

Usage

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

Arguments

model	(Model) Pharmpy model object
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 $\rightarrow$ 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 *deidentify_data*

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 preceeding 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 *display_odes*

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 *drop_columns*

Description

Drop columns from the dataset or mark as dropped

Usage

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

Arguments

model	(Model) Pharmpy model object
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) Pharmpy model object

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 object

Value

(Model) Pharmpy model object

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_modelfit_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_modelfit_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_modelfit_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 object
`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) Pharmpy model object

`filter_dataset` *filter_dataset*

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 object
`expr`	(str) expression for dataset query

Value

(Model) Pharmpy model object

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` *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` *fit*

Description

Fit models.

Usage

```
fit(model_or_models, esttool = NULL, path = NULL, context = NULL)
```

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
`path`	(str (optional)) Path to fit directory
`context`	(Context (optional)) Run in this context

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)
```

Arguments

`model`	(Model) Pharmpy model
`parameters`	(list(str=logical)) Set fix/unfix for these parameters

Value

(Model) Pharmpy model object

See Also

fix_parameters : Fix parameters

unfix_paramaters : Unfixing parameters

fix_paramaters_to : Fixing parameters and setting a new initial estimate in the same

function

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

function

Examples

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

## End(Not run)
```

`fix_parameters` *fix_parameters*

Description

Fix parameters

Fix all listed parameters

Usage

```
fix_parameters(model, parameter_names)
```

Arguments

`model`	(Model) Pharmpy model
`parameter_names`	(array(str) or str) one parameter name or a vector of parameter names

Value

(Model) Pharmpy model object

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)
```

Arguments

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

Value

(Model) Pharmpy model object

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$parameters['POP_CL']
model <- fix_parameters_to(model, {'POP_CL': 0.5})
model$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 *get_cmt*

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_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 object

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` *get_config_path*

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` *get_covariate_baselines*

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) Model to extract covariates from.

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` *get_dv_symbol*

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

(sympy.Symbol) 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 *get_evid*

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 *get_ids*

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`
get_individual_parameters

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

`model`	(Model) Pharmpy model to retrieve the individuals parameters from
`level`	(str) The variability level to look for: 'iiv', 'iov', 'random' or 'all' (default)
`dv`	(str or Expr or numeric (optional)) Name or DVID of dependent variable. NULL for all (default)

Value

(vectorc(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, 'iov')

## 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

`model`	(Model) Pharmpy model object

Value

(Expression) Symbolic expression

See Also

get_population_prediction_expression : Get full symbolic epression 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) Pharmpy 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` *get_lag_times*

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 *get_mdv*

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

Usage

```
get_model_code(model)
```

Arguments

model	(Model) Pharmpy model

Value

(str) Model code

Examples

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

Arguments

`model`	(Model) Pharmpy 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.

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 object

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 object

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 to retrieve parameters from
`parameter`	(str) Name of parameter to retrieve random variable from
`var_type`	(str) Variability type: iiv (default) or iov

Value

(vectorc(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 to retrieve the PD parameters from

Value

(vectorc(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 to retrieve the PK parameters from
`kind`	(str) The type of parameter to retrieve: 'absorption', 'distribution', 'elimination', or 'all' (default).

Value

(vectorc(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 object

Value

(Expression) Symbolic expression

See Also

get_individual_prediction_expression : Get full symbolic epression 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 to retrieve parameters from
`rv`	(str) Name of random variable to retrieve

Value

(vectorc(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 object

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 object

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)
```

`get_unit_of` *get_unit_of*

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 olumn, a parameter or a random variable.

Usage

```
get_unit_of(model, variable)
```

Arguments

`model`	(Model) Pharmpy model object
`variable`	(str) Find physical unit of this variable

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)
```

get_zero_order_inputs *get_zero_order_inputs*

Description

Get zero order inputs for all compartments

Usage

```
get_zero_order_inputs(model)
```

Arguments

model	(Model) Pharmpy model

Value

(sympy.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) Pharmpy model object

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) The model to check
`dv`	(str or Expr 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) The model to check
`dv`	(str or Expr 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 to check for covariate effect.
`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

(Bool : 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

(Bool : 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 Michalis-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_odes` *has_odes*

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

has_presystemic_metabolite

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) The model to check
`dv`	(str or Expr 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) Input 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) DPharmpy model

See Also

has_zero_order_absorption

has_first_order_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) The model to check

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

(Model) Reference to same model

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)
```

`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 = "same"
)
```

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_linearized` *is_linearized*

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)
```

`is_real` *is_real*

Description

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

Usage

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

Arguments

`model`	(Model) Pharmpy model
`expr`	(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)
```

---

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)
```

`list_time_varying_covariates` *list_time_varying_covariates*

Description

Return a vector of names of all time varying covariates

Usage

```
list_time_varying_covariates(model)
```

Arguments

`model`	(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` *load_dataset*

Description

Load the dataset given datainfo

Usage

```
load_dataset(model)
```

Arguments

`model`	(Model) Pharmpy model

Value

(Model) Pharmpy model with dataset removed

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) Loaded model object

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` *make_declarative*

Description

Make the model statments declarative

Each symbol will only be declared once.

Usage

```
make_declarative(model)
```

Arguments

`model`	(Model) Pharmpy model

Value

(Model) Pharmpy model object

Examples

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

## 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 object

Value

(Model) Pharmpy model object

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

`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) 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) 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) 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) 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) Previously run 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

Usage

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

Arguments

`model`	(Model) Previously run 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
)
```

Arguments

`model`	(Model) Previously run Pharmpy model.
`parameter_estimates`	(array or list(str=numeric)) Parameter estimates of model fit
`individual_estimates`	(data.frame) Individual estimates for etas

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`	(data.frame or str) 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 object
`modelfit_results`	(ModelfitResults) Pharmpy ModelfitResults object

`print_model_code` *print_model_code*

Description

Print the model code of the underlying model language

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 object

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)
```

Arguments

`path`	(str) Path to model

Value

(Model) Read model object

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 object

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) Pharmpy model object

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)
```

Arguments

`model`	(Model) Pharmpy model from which to remove the covariate effect.
`parameter`	(str) Name of parameter.
`covariate`	(str) Name of covariate.

Value

(Model) Pharmpy model object

Examples

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

## End(Not run)
```

`remove_derivative` *remove_derivative*

Description

Remove a derivative currently being calculcate when running model. Currently, only derivatives with respect to the prediction is supported. Default is to remove all that are present, First order derivates 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 varibles 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 modeas.
`with_respect_to`	(array(array(str) or str) or str (optional)) Parameter name(s) to use as independent variables. Default is NULL.

Value

(Pharmpy model.)

`remove_error_model` *remove_error_model*

Description

Remove error model.

Usage

```
remove_error_model(model)
```

Arguments

`model`	(Model) Remove error model for this model

Value

(Model) Pharmpy model object

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) Pharmpy model object

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

`model`	(Model) Pharmpy model to create block effect on.
`to_remove`	(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) Pharmpy model object

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` *remove_iov*

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 IOV from.
`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) Pharmpy model object

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) Pharmpy model object

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 object
`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 (optional)) Number of loq records to keep for each run of consecutive loq records.

Value

(Model) Pharmpy model object

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:

== == n == ================================
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) Name of compartment to remove peripheral compartment from.

Value

(Model) Pharmpy model object

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.

Usage

```
remove_predictions(model, to_remove = "all")
```

Arguments

`model`	(Model) Pharmpy model
`to_remove`	(array(str)) List of predictions to remove

Value

(Model) Pharmpy model object

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, 'all')
model$execution_steps[-1].predictions

## End(Not run)
```

remove_residuals *remove_residuals*

Description

Remove predictions and/or residuals

Remove residuals from estimation step.

Usage

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

Arguments

`model`	(Model) Pharmpy model
`to_remove`	(array(str)) List of predictions to remove

Value

(Model) Pharmpy model object

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, 'all')
model$execution_steps[-1].residuals

## 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 object

Value

(Model) Pharmpy model object

`rename_symbols` *rename_symbols*

Description

Rename symbols in the model

Make sure that no name clash occur.

Usage

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

Arguments

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

Value

(Model) Pharmpy model object

`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) A new 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 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` *Reset index*

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_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 model objects

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

Arguments

`model`	(Model (optional)) Pharmpy model
`results`	(ModelfitResults (optional)) 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,
  path = NULL,
  resume = FALSE,
  strictness = "minimization_successful or (rounding_errors and sigdigs>=0.1)",
  dv_types = NULL,
  mechanistic_covariates = NULL,
  retries_strategy = "all_final",
  seed = NULL,
  parameter_uncertainty_method = NULL,
  ignore_datainfo_fallback = FALSE
)
```

Arguments

`input`	(Model or str) Read model object/Path to a dataset
`results`	(ModelfitResults (optional)) Reults 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'. Default is 'default'
`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_blq` 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
`occasion`	(str (optional)) Name of occasion column
`path`	(str (optional)) Path to run AMD in
`resume`	(logical) Whether to allow resuming previous run
`strictness`	(str (optional)) 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'.
`seed`	(numeric (optional)) Random number generator or seed to be used.
`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

Value

(Model) Reference to the same model object

See Also

run_iiv

run_tool

Examples

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

## End(Not run)
```

---

run_bootstrap *run_bootstrap*

Description

Run bootstrap tool

Usage

```
run_bootstrap(model, results = NULL, resamples = 1, ...)
```

Arguments

`model`	(Model) Pharmpy model
`results`	(ModelfitResults (optional)) Results for model
`resamples`	(numeric) Number of bootstrap resample
`...`	Arguments to pass to tool

Value

(BootstrapResults) Bootstrap tool result object

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
run_bootstrap(model, res, resamples=500)

## End(Not run)
```

run_covsearch *run_covsearch*

Description

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

Usage

```
run_covsearch(
  search_space,
  p_forward = 0.01,
  p_backward = 0.001,
  max_steps = -1,
  algorithm = "scm-forward-then-backward",
  results = NULL,
  model = NULL,
  max_eval = FALSE,
  adaptive_scope_reduction = FALSE,
  strictness = "minimization_successful or (rounding_errors and sigdigs>=0.1)",
  naming_index_offset = 0,
  ...
)
```

Arguments

`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.
`results`	(ModelfitResults (optional)) Results of model
`model`	(Model (optional)) Pharmpy model
`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 (optional)) Strictness criteria
`naming_index_offset`	(numeric (optional)) index offset for naming of runs. Default is 0
`...`	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(search_space, model=model, results=results)

## 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` *run_iivsearch*

Description

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

Usage

```
run_iivsearch(
  algorithm = "top_down_exhaustive",
  iiv_strategy = "no_add",
  rank_type = "bic",
  linearize = FALSE,
  cutoff = NULL,
  results = NULL,
  model = NULL,
  keep = c("CL"),
  strictness = "minimization_successful or (rounding_errors and sigdigs>=0.1)",
  correlation_algorithm = NULL,
  E_p = NULL,
  E_q = NULL,
  ...
)
```

Arguments

`algorithm`	(str) Which algorithm to run when determining number of IIVs.
`iiv_strategy`	(str) If/how IIV should be added to start model. Default is 'no_add'.
`rank_type`	(str) Which ranking type should be used. Default is BIC.
`linearize`	(logical) Wheter 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)
`results`	(ModelfitResults (optional)) Results for model
`model`	(Model (optional)) Pharmpy model
`keep`	(array(str) (optional)) List of IIVs to keep. Default is "CL"
`strictness`	(str (optional)) 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 (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 (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 Non
`...`	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('td_brute_force', results=results, model=model)

## End(Not run)
```

run_iovsearch *run_iovsearch*

Description

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

Usage

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

Arguments

`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)
`results`	(ModelfitResults (optional)) Results for model
`model`	(Model (optional)) Pharmpy model
`strictness`	(str (optional)) Strictness criteria
`E`	(numeric (optional)) Expected number of predictors (used for mBIC). Must be set when using mBI
`...`	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('OCC', results=results, model=model)

## End(Not run)
```

---

run_linearize *run_linearize*

Description

Run linaerization procedure

Usage

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

Arguments

model	(Model (optional)) Pharmpy model.
model_name	(str) New name of linearized model. The default is "linbase".
description	(str) Description of linaerized model. The default is ""
...	Arguments to pass to tool

Value

(LinearizeResults) Linaerize tool results object.

---

run_modelfit *run_modelfit*

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_modelsearch` *run_modelsearch*

Description

Run Modelsearch tool. For more details, see :ref:`modelsearch`.

Usage

```
run_modelsearch(
  search_space,
  algorithm,
  iiv_strategy = "absorption_delay",
  rank_type = "bic",
  cutoff = NULL,
  results = NULL,
  model = NULL,
 strictness = "minimization_successful or (rounding_errors and sigdigs >= 0.1)",
  E = NULL,
  ...
)
```

Arguments

`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)
`results`	(ModelfitResults (optional)) Results for model
`model`	(Model (optional)) Pharmpy model
`strictness`	(str (optional)) Strictness criteria
`E`	(numeric (optional)) Expected number of predictors (used for mBIC). Must be set when using mBI
`...`	Arguments to pass to tool

Value

(ModelSearchResults) Modelsearch tool result object

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_results("pheno")
run_modelsearch('ABSORPTION(ZO);PERIPHERALS(1)', 'exhaustive', results=results, model=model)

## 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 = "",
  seed = 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 (optional)) 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.
`seed`	(numeric (optional)) Random number generator or seed to be used
`...`	Arguments to pass to tool

Value

(RetriesResults) Retries tool results object.

run_ruvsearch *run_ruvsearch*

Description

Run the ruvsearch tool. For more details, see :ref:`ruvsearch`.

Usage

```
run_ruvsearch(
  model = NULL,
  results = NULL,
  groups = 4,
  p_value = 0.001,
  skip = NULL,
  max_iter = 3,
  dv = NULL,
  strictness = "minimization_successful or (rounding_errors and sigdigs>=0.1)",
  ...
)
```

Arguments

`model`	(Model (optional)) Pharmpy model
`results`	(ModelfitResults (optional)) 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 (optional)) Strictness criteri
`...`	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` *run_simulation*

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` *run_structsearch*

Description

Run the structsearch tool. For more details, see :ref:`structsearch`.

Usage

```
run_structsearch(
  type,
  search_space = NULL,
  b_init = NULL,
  emax_init = NULL,
  ec50_init = NULL,
  met_init = NULL,
  results = NULL,
  model = NULL,
  extra_model = NULL,
  strictness = "minimization_successful or (rounding_errors and sigdigs >= 0.1)",
  extra_model_results = NULL,
  dv_types = NULL,
  ...
)
```

Arguments

`type`	(str) Type of model. Currently only 'drug_metabolite' and 'pkpd'
`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).
`results`	(ModelfitResults (optional)) Results for the start model
`model`	(Model (optional)) Pharmpy start model
`extra_model`	(Model (optional)) Optional extra Pharmpy model to use in TMDD structsearch
`strictness`	(str (optional)) Results for the extra model
`extra_model_results`	(ModelfitResults (optional)) Strictness criteria
`dv_types`	(list(str=numeric) (optional)) Dictionary of DV types for TMDD models with multiple DV
`...`	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_type='pkpd', results=results, model=model)

## 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(name, ...)
```

Arguments

`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)
```

---

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 = NULL
)
```

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 (optional)) 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 = NULL
)
```

Arguments

`model`	(Model) Input model
`parameter_estimates`	(array) 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 (optional)) 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 estiates given their covariance

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_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 ± estimate * fraction.

Usage

```
sample_parameters_uniformly(
  model,
  parameter_estimates,
  fraction = 0.1,
  force_posdef_samples = NULL,
  n = 1,
  seed = NULL,
  scale = "normal"
)
```

Arguments

`model`	(Model) Pharmpy model
`parameter_estimates`	(array) 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 (optional)) 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 estiates given their covariance

Examples

```
## Not run:
model <- load_example_model("pheno")
results <- load_example_modelfit_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) Set error model for this model
`dv`	(str or Expr 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) Pharmpy model object

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)
```

set_baseline_effect *set_baseline_effect*

Description

Create baseline effect model.

Currently implemented baseline effects are:

Constant baseline effect (const):

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

Usage

```
set_baseline_effect(model, expr = "const")
```

Arguments

model	(Model) Pharmpy model
expr	(str) Name of baseline effect function.

Value

(Model) Pharmpy model object

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_baseline_effect(model, expr='const')
model$statements$find_assignment("E")

## End(Not run)
```

`set_combined_error_model`
set_combined_error_model

Description

Set a combined error model. Initial estimates for new sigmas are (equation could not be rendered, see API doc on website) proportional and 0.09 for additive.

The error function being applied depends on the data transformation.

+————————+——————————————————+ | Data transformation | Combined error | +======================+===
| (equation could not be rendered, see API doc on website) +————————+———————————————————————————+ | (equation could not be rendered, see API doc on website) +——————————+——————————————————+

Usage

```
set_combined_error_model(model, dv = NULL, data_trans = NULL)
```

Arguments

`model`	(Model) Set error model for this model
`dv`	(str or Expr 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) Pharmpy model object

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` *set_covariates*

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)) List of column names

Value

(Model) Pharmpy model object

`set_dataset` *set_dataset*

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) Pharmpy model with new dataset and updated datainfo

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_direct_effect   *set_direct_effect*

Description

Add an effect to a model.

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)
```

Arguments

`model`	(Model) Pharmpy model
`expr`	(str) Name of PD effect function.

Value

(Model) Pharmpy model object

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) Pharmpy model object

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) Pharmpy model object

`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) Pharmpy model object

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 estimation step

Sets estimation step for a model. Methods currently supported are: FO, FOCE, ITS, LAPLACE, IMPMAP, IMP, SAEM, BAYES

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) Pharmpy model object

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) Model to set or change to use first order absorption rate

Value

(Model) Pharmpy model object

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) Pharmpy model object

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 to apply IIV on epsilons.
`dv`	(str or 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) Pharmpy model object

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) Pharmpy model object

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)
```

Arguments

`model`	(Model) Pharmpy model to update initial estimates
`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.

Value

(Model) Pharmpy model object

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_modelfit_results("pheno")
model$parameters$inits
model <- set_initial_estimates(model, results$parameter_estimates)
model$parameters$inits
model <- load_example_model("pheno")
model <- set_initial_estimates(model, {'POP_CL': 2.0})
model$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) Model to set or change absorption rate

Value

(Model) Pharmpy model object

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` *set_lloq_data*

Description

Set a dv value for lloq data records

Usage

```
set_lloq_data(model, value, lloq = NULL, blq = NULL)
```

Arguments

`model`	(Model) Pharmpy model object
`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) Pharmpy model object

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)
```

Arguments

model	(Model) Pharmpy model
bounds	(list(str=numeric)) A list of parameter bounds for parameters to change

Value

(Model) Pharmpy model object

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) Pharmpy model object

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) Pharmpy model object

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` *set_name*

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) Pharmpy model object

Examples

```
## Not run:
model <- load_example_model("pheno")
model$name
model <- set_name(model, "run2")
model$name

## End(Not run)
```

`set_ode_solver` *set_ode_solver*

Description

Sets ODE solver to use for model

Recognized solvers and their corresponding NONMEM advans:

+———————————-+——————————+ | Solver | NONMEM ADVAN | +============================+======
| CVODES | ADVAN14 | +———————————-+——————————+ | DGEAR | ADVAN8 | +——
———————————-+——————————+ | DVERK | ADVAN6 | +———————————-+——————
—+ | IDA | ADVAN15 | +———————————-+——————————+ | LSODA | ADVAN13 | +——
———————————-+——————————+ | LSODI | ADVAN9 | +———————————-+——————
—+

Usage

```
set_ode_solver(model, solver)
```

Arguments

model	(Model) Pharmpy model
solver	(str) Solver to use or NULL for no preference

Value

(Model) Pharmpy model object

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) Name of compartment to add peripheral to.

Value

(Model) Pharmpy model object

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_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 to create block effect on.
`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`	(str or Expr or numeric (optional)) Name or DVID of dependent variable. NULL will change the epsilon on all occurences 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) Pharmpy model object

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_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) Set error model for this model
`dv`	(str or Expr 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) Pharmpy model object

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 object
`refs`	(list(str=numeric)) Pairs of column names and reference values

Value

(Model) Pharmpy model object

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_reference_values(model, {'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) Model to set or change absorption rate

Value

(Model) Pharmpy model object

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` *set_simulation*

Description

Change model into simulation model

Usage

```
set_simulation(model, n = 1, seed = 64206)
```

Arguments

`model`	(Model) Pharmpy model
`n`	(numeric) Number of replicates
`seed`	(numeric) Random seed for the simulation

Value

(Model) Pharmpy model object

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 *set_time_varying_error_model*

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

Value

(Model) Pharmpy model object

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. dv_types = list('drug' = 1, 'target'= 2)). Default is NULL which means that all observations are treated as drug observations. For dv = 1 the only allowed keys are 'drug' and 'drug_tot'. If no DV for drug is specified then (free) drug will have dv = 1.

Value

(Model) Pharmpy model object

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 absorption rate is set the previous rate 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) Pharmpy model object

See Also

add_lag_time

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- set_transit_compartments(model, 3)
model$statements$ode_system

## End(Not run)
```

`set_upper_bounds` *set_upper_bounds*

Description

Set parameter upper bounds

Usage

```
set_upper_bounds(model, bounds)
```

Arguments

`model`	(Model) Pharmpy model
`bounds`	(list(str=numeric)) A list of parameter bounds for parameters to change

Value

(Model) Pharmpy model object

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_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) Pharmpy model object

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) Model to set or change to first order absorption rate

Value

(Model) Pharmpy model object

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) Pharmpy model object

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) Pharmpy model object

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 object
`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) Pharmpy model object

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 to separate. If NULL, all etas that are IIVs and non-fixed will become single. NULL is default.

Value

(Model) Pharmpy model object

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 estmation 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 option | LLOQ column | BLQ column | Used as indicator | Used as level | Note | +=============+=============+=========
| Available | NA | NA | DV < lloq | lloq | | +————-+————-+————+—————-+—
————+——————-+ | NA | Available | NA | DV < LLOQ | LLOQ | | +————-+——
——-+————+—————-+—————+——————-+ | NA | NA | Available | BLQ | nothing | Only for M1 and M7| +————-+————-+————+—————-+—————
+——————-+ | NA | NA | NA | NA | NA | No BLQ handling | +————-+————-+——
——+—————-+—————+—————-+ | NA | Available | Available | BLQ | LLOQ | DV column not used| +————-+————-+————+—————-+—————+———
———-+ | Available | NA | Available | BLQ | lloq | | +————-+————-+————+———
———-+—————+——————-+ | Available | Available | NA | DV < lloq | 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 DV 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 Pharmacokinet Pharmacodyn. 2001 Oct;28(5):481-504. doi: 10.1023/a:1012299115260. Erratum in: J Pharmacokinet 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) Pharmpy model object

See Also

remove_loq_data

set_lloq_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 to apply boxcox transformation to.
`list_of_etas`	(array(str) or str (optional)) Name/names of etas to transform. If NULL, all etas will be transformed (default).

Value

(Model) Pharmpy model object

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 (1) to spelected etas

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 to apply John Draper transformation to.
`list_of_etas`	(array(str) or str (optional)) Name/names of etas to transform. If NULL, all etas will be transformed (default).

Value

(Model) Pharmpy model object

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 to apply t distribution transformation to.
`list_of_etas`	(array(str) or str (optional)) Name/names of etas to transform. If NULL, all etas will be transformed (default).

Value

(Model) Pharmpy model object

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 object

Value

(Model) Pharmpy model object

`unconstrain_parameters` *unconstrain_parameters*

Description

Remove all constraints from parameters

Usage

```
unconstrain_parameters(model, parameter_names)
```

Arguments

`model`	(Model) Pharmpy model
`parameter_names`	(array(str)) Remove all constraints for the listed parameters

Value

(Model) Pharmpy model object

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 *undrop_columns*

Description

Undrop columns of model

Usage

```
undrop_columns(model, column_names)
```

Arguments

`model`	(Model) Pharmpy model object
`column_names`	(array(str) or str) List of column names or one column name to undrop

Value

(Model) Pharmpy model object

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 *unfix_parameters*

Description

Unfix parameters

Unfix all listed parameters

Usage

```
unfix_parameters(model, parameter_names)
```

Arguments

`model`	(Model) Pharmpy model
`parameter_names`	(array(str) or str) one parameter name or a vector of parameter names

Value

(Model) Pharmpy model object

See Also

unfix_paramaters_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)
```

Arguments

`model`	(Model) Pharmpy model
`inits`	(list(str=numeric)) Inits for all parameters to unfix and change init

Value

(Model) Pharmpy model object

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, {'POP_CL': 0.5})
model$parameters$fix
model$parameters['POP_CL']

## End(Not run)
```

`unload_dataset` *unload_dataset*

Description

Unload the dataset from a model

Usage

```
unload_dataset(model)
```

Arguments

`model`	(Model) Pharmpy model

Value

(Model) Pharmpy model with dataset removed

Examples

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

## End(Not run)
```

`update_initial_individual_estimates` *update_initial_individual_estimates*

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 to update initial estimates
`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) Pharmpy model object

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) Pharmpy model object

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

`model`	(Model) Model whose dataset to write to file
`path`	(str (optional)) Destination path. Default is to use original path with .csv suffix.
`force`	(logical) Overwrite file with same path. Default is FALSE.

Value

(Model) Updated model object

Examples

```
## Not run:
model <- load_example_model("pheno")
model <- write_csv(model, path="newdataset$csv")

## End(Not run)
```

`write_model` *write_model*

Description

Write model code to file

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) Pharmpy model object

Examples

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

## End(Not run)
```

`write_results` *write_results*

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, 7
add_allometry, 8
add_bioavailability, 9
add_cmt, 10
add_covariate_effect, 11
add_derivative, 13
add_effect_compartment, 14
add_estimation_step, 15
add_iiv, 16
add_indirect_effect, 18
add_individual_parameter, 19
add_iov, 20
add_lag_time, 21
add_metabolite, 22
add_parameter_uncertainty_step, 22
add_pd_iiv, 23
add_peripheral_compartment, 24
add_pk_iiv, 25
add_population_parameter, 26
add_predictions, 27
add_residuals, 28
add_time_after_dose, 29
append_estimation_step_options, 30

bin_observations, 31
bump_model_number, 32

calculate_aic, 32
calculate_bic, 33
calculate_corr_from_cov, 34
calculate_corr_from_prec, 35
calculate_cov_from_corrse, 36
calculate_cov_from_prec, 37
calculate_epsilon_gradient_expression, 38
calculate_eta_gradient_expression, 38
calculate_eta_shrinkage, 39
calculate_individual_parameter_statistics, 40
calculate_individual_shrinkage, 41
calculate_parameters_from_ucp, 42
calculate_pk_parameters_statistics, 43
calculate_prec_from_corrse, 44
calculate_prec_from_cov, 45
calculate_se_from_cov, 46
calculate_se_from_prec, 47
calculate_ucp_scale, 48
check_dataset, 48
check_high_correlations, 49
check_parameters_near_bounds, 50
check_pharmpy, 51
cleanup_model, 51
convert_model, 52
create_basic_pk_model, 53
create_config_template, 54
create_joint_distribution, 54
create_report, 55
create_rng, 55
create_symbol, 56

deidentify_data, 57
display_odes, 57
drop_columns, 58
drop_dropped_columns, 59

evaluate_epsilon_gradient, 59
evaluate_eta_gradient, 60
evaluate_expression, 61
evaluate_individual_prediction, 62
evaluate_population_prediction, 63
evaluate_weighted_residuals, 64
expand_additional_doses, 65

filter_dataset, 65
find_clearance_parameters, 66
find_volume_parameters, 67
fit, 67
fix_or_unfix_parameters, 68
fix_parameters, 69
fix_parameters_to, 70

get_admid, 71
get_baselines, 71
get_bioavailability, 72
get_central_volume_and_clearance, 72
get_cmt, 73
get_concentration_parameters_from_data, 73
get_config_path, 74
get_covariate_baselines, 74
get_covariate_effects, 75
get_doseid, 76
get_doses, 76
get_dv_symbol, 77
get_evid, 78
get_ids, 78
get_individual_parameters, 79
get_individual_prediction_expression, 80
get_initial_conditions, 80
get_lag_times, 81
get_mdv, 82
get_model_code, 82
get_model_covariates, 83
get_number_of_individuals, 83
get_number_of_observations, 84
get_number_of_observations_per_individual, 85
get_number_of_peripheral_compartments, 86
get_number_of_transit_compartments, 86
get_observation_expression, 87
get_observations, 87
get_omegas, 88
get_parameter_rv, 89
get_pd_parameters, 90
get_pk_parameters, 90
get_population_prediction_expression, 91
get_rv_parameters, 92
get_sigmas, 93
get_thetas, 93
get_unit_of, 94
get_zero_order_inputs, 95
greekify_model, 95

has_additive_error_model, 96
has_combined_error_model, 97
has_covariate_effect, 98
has_first_order_absorption, 98
has_first_order_elimination, 99
has_instantaneous_absorption, 100
has_linear_odes, 100
has_linear_odes_with_real_eigenvalues, 101
has_michaelis_menten_elimination, 102
has_mixed_mm_fo_elimination, 102
has_odes, 103
has_presystemic_metabolite, 104
has_proportional_error_model, 104
has_random_effect, 105
has_seq_zo_fo_absorption, 106
has_weighted_error_model, 107
has_zero_order_absorption, 107
has_zero_order_elimination, 108

install_pharmpy, 109
install_pharmpy_devel, 109
is_linearized, 110
is_real, 110
is_strictness_fulfilled, 111

list_time_varying_covariates, 112
load_dataset, 112
load_example_model, 113
load_example_modelfit_results, 114

make_declarative, 114
mu_reference_model, 115

omit_data, 116

plot_abs_cwres_vs_ipred, 116
plot_cwres_vs_idv, 117
plot_dv_vs_ipred, 118
plot_dv_vs_pred, 119
plot_eta_distributions, 119
plot_individual_predictions, 120
plot_iofv_vs_iofv, 121
plot_transformed_eta_distributions, 122
plot_vpc, 122
predict_influential_individuals, 123
predict_influential_outliers, 124
predict_outliers, 125
print_fit_summary, 126
print_model_code, 126
print_model_symbols, 127
print_pharmpy_version, 127

read_dataset_from_datainfo, 128
read_model, 128
read_model_from_string, 129
read_modelfit_results, 129
read_results, 130
remove_bioavailability, 131
remove_covariate_effect, 131
remove_derivative, 132
remove_error_model, 133
remove_estimation_step, 133
remove_iiv, 134
remove_iov, 135
remove_lag_time, 136
remove_loq_data, 137
remove_parameter_uncertainty_step, 138
remove_peripheral_compartment, 139
remove_predictions, 140
remove_residuals, 141
remove_unused_parameters_and_rvs, 142
rename_symbols, 142
replace_non_random_rvs, 143
resample_data, 143
reset_index, 144
reset_indices_results, 145
retrieve_models, 145
run_allometry, 146
run_amd, 147
run_bootstrap, 149
run_covsearch, 150
run_estmethod, 151
run_iivsearch, 152
run_iovsearch, 154
run_linearize, 155
run_modelfit, 155
run_modelsearch, 156
run_retries, 157
run_ruvsearch, 158
run_simulation, 159
run_structsearch, 160
run_tool, 161

sample_individual_estimates, 162
sample_parameters_from_covariance_matrix, 163
sample_parameters_uniformly, 164
set_additive_error_model, 166
set_baseline_effect, 167
set_combined_error_model, 168
set_covariates, 169
set_dataset, 169
set_direct_effect, 170
set_dtbs_error_model, 171
set_dvid, 172
set_estimation_step, 172
set_evaluation_step, 173
set_first_order_absorption, 174
set_first_order_elimination, 175
set_iiv_on_ruv, 176
set_initial_condition, 177
set_initial_estimates, 177
set_instantaneous_absorption, 178
set_lloq_data, 179
set_lower_bounds, 180
set_michaelis_menten_elimination, 181
set_mixed_mm_fo_elimination, 182
set_name, 183
set_ode_solver, 183
set_peripheral_compartments, 184
set_power_on_ruv, 185
set_proportional_error_model, 186
set_reference_values, 187
set_seq_zo_fo_absorption, 188
set_simulation, 189
set_time_varying_error_model, 190
set_tmdd, 190
set_transit_compartments, 191
set_upper_bounds, 192
set_weighted_error_model, 193
set_zero_order_absorption, 194
set_zero_order_elimination, 194
set_zero_order_input, 195
simplify_expression, 196
solve_ode_system, 197
split_joint_distribution, 197
summarize_modelfit_results, 198

transform_blq, 199
transform_etas_boxcox, 200
transform_etas_john_draper, 201
transform_etas_tdist, 202
translate_nmtran_time, 203

unconstrain_parameters, 203
undrop_columns, 204
unfix_parameters, 205
unfix_parameters_to, 206
unload_dataset, 207

update_initial_individual_estimates,
207
use_thetas_for_error_stdev, 208

write_csv, 209
write_model, 209
write_results, 210