Package 'SimuRg'

August 24, 2026

Title Building, Fitting and Evaluating PK/PD Modeles

Version 0.2.2

Description Provides a unified workflow for building, fitting using external engines, and evaluating ordinary differential equation (ODE)-based pharmacokinetic/pharmacodynamic (PK/PD) models. Supports generation of estimation scenarios and control files for external engines (e.g., 'Monolix'), simulation of models using 'rxode2', and creation of goodness-of-fit diagnostics. Includes tools for covariate modeling, virtual population design, and local and global sensitivity analyses.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

Imports cluster, dplyr, fastDummies, forcats, ggplot2, jsonlite, MASS, purrr, readr, recipes, rlang, rxode2, scales, stringr, synthpop, sys, tibble, tidyr, uwot, zoo, lhs, sensitivity, ppcor, withr

Config/testthat/edition 3

Depends R (>= 3.5)

LazyData true

URL https://ms-decisions.github.io/SimuRg/, https://github.com/ms-decisions/SimuRg

BugReports https://github.com/ms-decisions/SimuRg/issues

VignetteBuilder knitr

NeedsCompilation no

Author Victor Sokolov [cph, aut] (ORCID: <https://orcid.org/0000-0001-7066-0811>), Anna Mikhailova [cre, aut] (ORCID: <https://orcid.org/0009-0007-1547-5200>), Yaroslav Ugolkov [aut] (ORCID: <https://orcid.org/0000-0003-3590-7860>), Anatoly Pokladyuk [aut] (ORCID:

<https://orcid.org/0009-0003-4680-9518>),
Alina Melnikova [aut] (ORCID: <https://orcid.org/0000-0001-9019-0263>),
Victoria Kulesh [aut] (ORCID: <https://orcid.org/0000-0001-9289-7262>)

Maintainer Anna Mikhailova <anna.mikhailova@msdecisions.tech>

Repository CRAN

Date/Publication 2026-08-24 12:50:08 UTC

Contents

`data_pbc` *Primary Biliary Cirrhosis (PBC) dataset*

Description

Clinical trial dataset with Primary Biliary Cirrhosis patient data including survival outcomes and covariates.

Usage

```
data_pbc
```

Format

`data_pbc`**:**

A data frame with 280 rows and 15 columns:

id identifier of the individual

years time in years (survival time)

status2 survival status (0 = censored, 1 = event)

drug treatment group ("D-penicil" or "placebo")

age age of the individual in years

sex sex of the individual ("male" or "female")

ascites presence of ascites ("Yes" or "No")

hepatomegaly presence of hepatomegaly ("Yes" or "No")

spiders presence of spider angiomas ("Yes" or "No")

edema edema status ("No edema", "edema no diuretics", or "edema despite diuretics")

serBilir serum bilirubin level (mg/dl)

serChol serum cholesterol level (mg/dl)

albumin albumin level (g/dl)

platelets platelet count (per cubic mm/1000)

ds_covval *Warfarin covariate values dataset*

Description

Individual covariate values for 100 patients used in the warfarin covariate sensitivity simulation, including demographic and pharmacogenomic variables.

Usage

```
ds_covval
```

Format

`ds_covval`**:**

A data frame with 100 rows and 9 columns:

ID individual patient identifier

AGE age in years

WEIGHT body weight in kg

BMI body mass index (kg/m^2)

LG_WEIGHT log-transformed body weight (log10 scale), used as continuous covariate in the PK model

LG_AGE log-transformed age (log10 scale), used as continuous covariate in the PK model

SEX sex of the individual (0 = female, 1 = male)

CYP2C9 CYP2C9 genotype encoded as integer: 0 = **1/*1* (reference), 1 = **1/*2*, 2 = **1/*3*, 3 = **2/*2*, 4 = **2/*3*, 5 = **3/*3*

VKORC1 VKORC1 genotype (`"GG"`, `"AG"`, or `"AA"`)

Source

Derived from `data_fin_i.csv` — individual covariate dataset for the warfarin PopPK covariate sensitivity analysis.

GCO *Generalized Control Object (GCO)*

Description

Generalized control object (GCO) stores the model setup, dataset mapping, and estimation options used for fitting or downstream simulation configuration.

Format

GCO:

A named list with the following components:

headers List of lists. One element per dataset column with fields: name (character, column name), use (character, Monolix column role such as "identifier", "time", "observation", "covariate", etc.), type (character or empty; for covariates: "continuous" or "categorical").

data Character. Path to the source dataset file.

model Character. Path to the structural model file (Monolix syntax).

task_opt List. Task options passed to the fitter; empty list when not set.

covs Character vector. Names of covariate columns detected in the dataset headers.

project_name Character. Project or run name.

theta List of lists or data frame. Population (typical) parameter definitions. Each record contains: NAME (character, parameter name without _pop suffix), TRANS (character, "normal", "logNormal", or "logitNormal"), INIT (numeric, initial or fixed value on the natural scale), EST (logical, whether the parameter is estimated). When passed to sg_fit(), a tibble with optional LB and UB for logit bounds is also accepted.

ruv List, or list of lists for multiple observation types. Residual error model definition with fields: YNAME (character, longitudinal output name, e.g. "y1"), DVID (numeric, observation-type identifier), TRANS (character, residual distribution), PRED (character, prediction variable name), ERR (character, error model type such as "constant", "proportional", "combined1"), INIT (numeric vector of initial residual-error parameter values), EST (logical vector, estimation flags), BLQM (optional BLQ handling method; NULL when not used).

re List with init and est numeric matrices. Between-subject variability: diagonal elements are initial variances of omega_* parameters, off-diagonal elements are covariances; est uses TRUE/FALSE/NA in the same layout as sg_fit() input.

occ List with init and est matrices. Between-occasion variability in the same layout as re; all-zero/NA when occasion variability is not modeled.

modelText Character. Full text of the structural model file.

Details

Create a GCO with sg_converter() from a Monolix project, or with sg_fit().

See Also

GFO, GMO, sg_fit(), sg_converter(), read_smrg_ctrl()

Examples

```
# Bundled GCO (Simurg fit configuration, JSON format)
gco_path <- system.file("extdata", "simurg_object", "sg_object.json",
                        package = "SimuRg")
gco <- read_smrg_ctrl(gco_path)
names(gco)
gco$project_name
gco$theta
gco$ruv
```

```
gco$re$init
```

GFO *Generalized Fit Object (GFO)*

Description

Generalized fit object (GFO) stores population and individual estimation results, diagnostic tables, and variance–covariance matrices produced by model fitting.

Format

GFO**:**

A named list with the following components:

SDTAB Data frame. Standard diagnostic table of observations and model predictions. Typical columns: ID, TIME, DV (observed), DVID, DVNAME, PRED (population prediction), IPRED (individual prediction), RES, IRES, WRES, IWRES (residuals), MDV, and optionally OCC, BLQ, CENS, LIMIT.

SUMTAB Data frame. Parameter summary table. Typical columns: PAR (parameter name), VALUE (estimate), TYPE ("Typical values", "Random effects", "Covariate coefficients", "Correlation coefficients", or "Residual error model"), DISTRIBUTION, EST ("ESTIMATED" or "FIXED"), SE, RSE, CV, LCI, UCI, ETAshrinkage_var, ETAshrinkage_sd, EPSshrinkage_sd.

SIGMAMAT Numeric matrix. Residual error variance–covariance matrix (diagonal elements are residual variance parameters).

OMEGAMAT Numeric matrix. Inter-individual variability variance–covariance matrix on the omega_* scale.

OCCMAT Numeric matrix. Inter-occasion variability matrix; empty when occasion variability is not present.

EVTAB Data frame. Dosing/event table extracted from the dataset. Typical columns: ID, TIME, EVID, CMT, ADM, AMT, and optionally OCC, ADDL, II, DUR, TINF, RATE, SS.

PATAB Data frame. Individual random effects and post hoc parameter estimates. Columns ID, eta_* (individual ETAs), and individual PK/PD parameter columns (e.g. ka, Vd, CL).

COTAB Data frame. Continuous covariates: ID plus continuous covariate columns (including derived columns such as log-transformed covariates). Empty data frame when none are present.

CATAB Data frame. Categorical covariates: ID plus categorical covariate columns. Empty data frame when none are present.

REGTAB Data frame. Time-varying regressors (ID, TIME, regressor columns). Empty data frame when none are present.

OFV Data frame with one row. Model fit criteria parsed from Monolix summary.txt: LL (minus 2 times log-likelihood, Monolix OFV), AIC, BIC, BICc.

COVMAT Numeric matrix. Variance–covariance matrix of population parameter estimates.

CORRMAT Numeric matrix. Correlation matrix of population parameter estimates.

OPTIONS List or NULL. Additional model or task options when available.

PROJNAME Character. Project or run name.

Details

Create a GFO with `sg_converter()` from a Monolix project, or with `sg_fit()` when `fit = TRUE`.

See Also

GCO, `sg_fit()`, `sg_converter()`, `read_smrg_obj()`, gfo4cov

Examples

```
gfo <- read_smrg_obj(gfo4cov)
names(gfo)
head(gfo$SDTAB)
head(gfo$SUMTAB)
head(gfo$PATAB)
gfo$OFV
gfo$OMEGAMAT
```

---

gfo4cov *Warfarin PopPK model fitting output (4-covariate model)*

Description

A list containing the full output of a warfarin population pharmacokinetic model fit with four covariates (SEX, LG_WEIGHT, CYP2C9, VKORC1), including parameter estimates, model diagnostics, individual predictions, and covariate data.

Usage

```
gfo4cov
```

Format

`gfo4cov`**:**

A named list with 12 elements:

PROJNAME character string identifying the project run (`"run_4cov"`)

SUMTAB data frame with 15 rows and 11 columns containing the parameter summary table. Columns: `PAR` (parameter name), `VALUE` (estimate), `TYPE` (category: `"Typical values"`, `"Covariate effects"`, `"Random effects"`, or `"Residual error model"`), `EST` (estimation status), `SE` (standard error), `RSE` (relative SE, \ `LCI` and `UCI` (95\ `ETAshrinkage_var`, `ETAshrinkage_sd`, `EPSshrinkage_sd` (shrinkage metrics).

SDTAB data frame with 1600 rows and 11 columns — the standard table of observations and model predictions. Columns: `ID`, `TIME`, `DV` (observed), `DVID`, `PRED` (population prediction), `IPRED` (individual prediction), `RES` (residual), `IRES` (individual residual), `WRES` (weighted residual), `IWRES` (individual weighted residual), `MDV`.

PATAB data frame with 100 rows and 7 columns — individual parameter table. Columns: `ID`, `eta_ka`, `eta_Vd`, `eta_CL` (individual ETA random effects), `ka`, `Vd`, `CL` (individual post-hoc PK parameter estimates).

COVMAT 15×15 numeric matrix — covariance matrix of the population parameter estimates.

CORRMAT 15×15 numeric matrix — correlation matrix of the population parameter estimates.

OFV data frame with 1 row and 4 columns — model fit criteria: `LL` (`-2 x log-likelihood`, i.e. Monolix OFV), `AIC`, `BIC`, `BICc`.

OMEGAMAT 3×3 numeric matrix — OMEGA variance-covariance matrix of the inter-individual random effects (`eta_ka`, `eta_Vd`, `eta_CL`).

SIGMAMAT 1×1 numeric matrix — SIGMA residual error variance matrix.

EVTAB data frame with 100 rows and 6 columns — dosing/event table. Columns: `ID`, `TIME`, `EVID`, `AMT`, `ADM`, `CMT`.

COTAB data frame with 100 rows and 6 columns — continuous covariate table. Columns: `ID`, `AGE`, `WEIGHT`, `BMI`, `LG_WEIGHT` (log10-transformed weight), `LG_AGE` (log10-transformed age).

CATAB data frame with 100 rows and 4 columns — categorical covariate table. Columns: `ID`, `SEX` (0 = female, 1 = male), `CYP2C9` (genotype integer code, 0–5), `VKORC1` (genotype: `"GG"`, `"AG"`, or `"AA"`).

Source

Generated from the warfarin PopPK model fitting run `"run_4cov"` used in the covariate sensitivity analysis workflow.

`GMO` *Generalized Model Object (GMO)*

Description

Generalized model object (GMO) is an RxODE2 model that encodes the full pharmacometric model in simulation syntax: structural parameters, covariate effects, inter-individual variability, and residual error components.

Format

`GMO`**:**

An RxODE2 model object (`rxode2` class). Typical contents:

Typical values Fixed effects on the log scale (`ka_pop`, `Vd_pop`, `CL_pop`, ...) with transformation lines such as `ka_tv = exp(ka_pop)`.

Random effects Initial values for ETAs (`omega_ka`, `omega_Vd`, `omega_CL`, ...) and individual parameters such as `ka = ka_tv * exp(omega_ka)`.

Covariate effects Covariate coefficients (e.g. `beta_Vd_WT`) applied to typical values, e.g. `Vd = Vd_tv * exp(omega_Vd) * exp(beta_Vd_WT * WT)`.

Structural model Initial conditions and ODE equations (`Ad`, `Ac`, `Cc`, ...).

Residual error Error model parameters (e.g. `Cc_b`) and observation outputs such as `Cc_ResErr = Cc * (1 + Cc_b)`.

Details

Build a GMO with `rxode2::rxode2()` or use the bundled gmo_pk1c example.

See Also

GCO, GSI, gmo_pk1c, `sg_sim()`

Examples

```
gmo_pk1c_comp <- eval(parse(text = paste0("rxode2::rxode2({", gmo_pk1c, "})")))
gmo_pk1c_comp$model
gmo_pk1c_comp$params
```

---

gmo_pk1c *One-compartment PK generalized model object (GMO)*

Description

Bundled GMO: one-compartment oral PK model with inter-individual variability on `ka`, `Vd`, and `CL`, linear covariate effect of `WT` on `Vd` (`beta_Vd_WT`), and proportional residual error on `Cc_ResErr`.

Usage

```
gmo_pk1c
```

Format

`gmo_pk1c`**:**

An `rxode2` model object with population parameters (`*_pop`), random-effect placeholders (`omega_*`), covariate effect `beta_Vd_WT`, structural ODEs (`Ad`, `Ac`, `Cc`), and residual error output (`Cc_ResErr`).

GSI *Generalized Simulations Input (GSI)*

Description

Generalized simulations input (GSI) is the set of arguments passed to `sg_sim()` to define a simulation scenario: model, event table, parameter values, and optional uncertainty or variability matrices. See bundled gsi_pk1c for a complete example.

Format

GSI:

A named list (not a formal S3 class) with components matching sg_sim() arguments. When serialized to JSON, the same information may use output instead of outputs and store model as a character string; in R, pass an GMO object to sg_sim().

model GMO. RxODE2 model (gmo_pk1c in bundled examples).

et Data frame. Event table; columns such as ID, TIME, AMT, ADDL, II, and covariates (e.g. WT).

stimes Numeric vector. Output sampling time grid.

outputs Character vector. Model output names to retain (e.g. "Cc").

covs Character vector. Covariate column names taken from et.

theta Named numeric vector. Population parameters on the model scale (ka_pop, Vd_pop, CL_pop, covariate coefficients, ...).

thetamat Numeric matrix. Parameter-estimation covariance for population uncertainty (npop > 1).

omega, sigma Matrix or NULL. Inter-individual and residual variability; empty when not resampled.

npop, nsub Integers. Population and subject replicate counts.

byID, byPOP, shared, aggr, addcov, keep, ... Optional sg_sim() controls.

See Also

GSO, GMO, gmo_pk1c, gsi_pk1c, sg_sim()

Examples

```
names(gsi_pk1c)
head(gsi_pk1c$et)
gsi_pk1c$theta
gsi_pk1c$npop
# Pass to sg_sim() together with the GMO:
do.call(sg_sim, c(list(model = gmo_pk1c), gsi_pk1c))
```

---

gsi_pk1c *Generalized simulations input (GSI) for one-compartment PK example*

Description

Bundled GSI matching the standard simulation scenario: two dosing regimens with different WT, population parameter uncertainty (thetamat, npop = 10), and output Cc only.

Usage

```
gsi_pk1c
```

Format

`gsi_pk1c`:

A named list ready for `sg_sim()`, with components:

`et` Data frame. Event table (`ID`, `TIME`, `AMT`, `ADDL`, `II`, `WT`).

`stimes` Numeric vector. Output time grid (0–120 h).

`outputs` Character vector. Model outputs to retain (`"Cc"`).

`covs` Character vector. Covariate columns joined from `et` (`"WT"`).

`theta` Named numeric vector. Population parameters (`ka_pop`, `Vd_pop`, `CL_pop`, `beta_Vd_WT`).

`thetamat` Numeric matrix. Parameter estimation covariance (8 x 8).

`omega`, `sigma` `NULL` when inter-individual and residual variability are not resampled.

`npop`, `nsub` Integers. Numbers of population (`10`) and subject (`1`) replicates.

`addcov` Logical. `FALSE`; covariates are already in `et`.

`GSO` *Generalized Simulations Output (GSO)*

Description

Generalized simulations output (GSO) is the long-format data frame returned by `sg_sim()` and related simulation helpers. Run the bundled gsi_pk1c scenario with gmo_pk1c to reproduce the standard example output.

Format

`GSO`:

A data frame in long format with one row per `ID`, replicate, time point, and output variable. With population uncertainty (`thetamat`, `npop` > 1) typical columns are:

`ID` Dosing scenario identifier from the event table.

`POPN` Population replicate index (present when `thetamat` is used).

`sim.id` Subject replicate index (present when `omega` is used).

`TIME` Simulation time (h).

`VAR` Output variable name (e.g. `"Cc"`).

`VALUE` Simulated value.

When `aggr` is set, aggregation statistic columns (`mean`, `median`, `P05`, ...) replace or supplement `VALUE`; see the Details section of `sg_sim()`.

See Also

GSI, GMO, gmo_pk1c, gsi_pk1c, `sg_sim()`

Examples

```
gso <- do.call(sg_sim, c(list(model = gmo_pk1c), gsi_pk1c))
names(gso)
head(gso)
subset(gso, VAR == "Cc" & ID == 1 & POPN == 1)
```

`parest` *Warfarin population PK parameter estimates*

Description

Final parameter estimates from a warfarin population pharmacokinetic model, including typical values, covariate effects, random effects, and residual error parameters with associated uncertainty measures.

Usage

```
parest
```

Format

`parest`**:**

A data frame with 15 rows and 12 columns:

parameter parameter name as used in the model

value final parameter estimate

TYPE parameter category:`"Typical values"`, `"Covariate effects"`, `"Random effects"`, or `"Residual error model"`

EST estimation status (`"ESTIMATED"`)

SE standard error of the estimate

RSE relative standard error (%)

LCI lower bound of the 95 % confidence interval

UCI upper bound of the 95 % confidence interval

ETAshrinkage_var ETA shrinkage on the variance scale (%); `NA` for non-random-effect parameters

ETAshrinkage_sd ETA shrinkage on the SD scale (%); `NA` for non-random-effect parameters

EPSshrinkage_sd EPS shrinkage on the SD scale (%); `NA` for non-residual-error parameters

Source

Derived from `par_fin_i.csv` — warfarin PopPK model estimation output.

read_smrg_ctrl *Read or validate a generalized control object (GCO)*

Description

Loads a generalized control object from an `.RData` or `.json` file, or accepts an already-loaded list.

Usage

```
read_smrg_ctrl(ctrl)
```

Arguments

`ctrl`	Character path to a `.RData` or `.json` file, or a GCO list object.

Value

A GCO list. When `ctrl` is a file path, the object is loaded from disk; when `ctrl` is already a list, it is returned unchanged.

See Also

GCO, `read_smrg_obj()`, `sg_fit()`, `sg_converter()`

read_smrg_obj *Read or validate a generalized fit object*

Description

Loads a generalized fit object from an `.RData` or `.json` file, or accepts an already-loaded list and ensures all table components (`SDTAB`, `EVTAB`, `COTAB`, `CATAB`, `SUMTAB`) are proper data frames.

Usage

```
read_smrg_obj(fpath_i)
```

Arguments

`fpath_i`	String, GFO, or a named list with a `GFO` component (and optionally `GCO`). If a string is given, the path to a `.RData` or `.json` file with a fit object is expected.

Value

GFO, or a named list with `GFO` (and optionally `GCO`) components, with all table components coerced to data frames.

Examples

```
# Pass an already-loaded sg-fit object
obj <- read_smrg_obj(gfo4cov)
class(obj$SDTAB)
```

`sg_converter` *Convert Monolix project output to R objects*

Description

Reads and parses Monolix project output files (.mlxtran and associated data files) into a structured list of SimuRg objects including parameter estimates, individual predictions, residuals, and diagnostic information.

Usage

```
sg_converter(folder_path, proj_name, save_file = FALSE)
```

Arguments

`folder_path`	Character string. Path to the directory containing Monolix project files.
`proj_name`	Character string. Name of the Monolix project (without file extension).
`save_file`	Logical. If `TRUE`, saves `GCO` and `GFO` JSON files to `folder_path` with names `<proj_name>_GCO.json` and `<proj_name>_GFO.json`, and also saves two RData files: `<proj_name>_GCO.RData` (object `gco`) and `<proj_name>_GFO.RData` (object `gfo`).

Details

This function serves as a bridge between Monolix output and R by parsing the .mlxtran file and associated data files to create a comprehensive R object containing all relevant model outputs. This facilitates further analysis, visualization, and reporting in R.

The function automatically detects and imports various components of Monolix output including population parameters, individual parameters, covariates, and diagnostic metrics.

If `save_file = TRUE`, the function additionally writes GCO and GFO JSON files and `.RData` files to `folder_path`.

Value

Named list with GCO and GFO components.

See Also

GCO, GFO

Examples

```
library(stringr)
# Convert Monolix project results
test_folder <- system.file("extdata", "Monolix_objects", package = "SimuRg")
if (substr(test_folder, nchar(test_folder), nchar(test_folder)) != "/")
  test_folder <- str_c(test_folder, "/")
pro_name <- "proj-solo"
result <- sg_converter(folder_path = test_folder, proj_name = pro_name)
# save(results, file = "./models/simurg_object/Warfarin_PK.RData")
# Access individual predictions
head(result$GFO$SDTAB)

# View parameter estimates
print(result$GFO$SUMTAB)

# Check objective function value
print(result$GFO$OFV)
```

`sg_covsearch` *Stepwise covariate modeling (SCM) search*

Description

Runs forward inclusion and optional backward elimination of parameter–covariate relationships, selecting terms by likelihood-ratio tests against `p_forward` and `p_backward`.

Usage

```
sg_covsearch(
  gfo,
  gco,
  output_dir = NULL,
  covariates = NULL,
  parameters = NULL,
  test_pairs = NULL,
  p_forward = 0.05,
  p_backward = 0.01,
  fit_function = sg_fit,
  update_theta_init = FALSE,
  run_backward = TRUE,
  update_theta_init_backward = FALSE,
  path_to_fitter = NULL
)
```

Arguments

gfo	GFO or character. Baseline fit object, or path to a GFO `.json`/`.RData`, containing at least `COTAB` and `CATAB`.
gco	GCO or character. Control object (list), or path to a GCO `.json`/`.RData`, containing at least `headers` and `theta`.
output_dir	Character. Directory where fit projects and search history files are written. Default value is `tempdir()`.
covariates	Character vector or `NULL`. Covariate names to consider. When `NULL`, all covariates from `gco$headers` are used. Default value is `NULL`.
parameters	Character vector or `NULL`. Parameter names to consider. When `NULL`, all parameters from `gco$theta` are used. Default value is `NULL`.
test_pairs	Data.frame or `NULL`. Candidate pairs with columns `parameter`, `covariate`, `type`, `reference`, and `center`. When `NULL`, all parameter–covariate combinations are generated. Default value is `NULL`.
p_forward	Numeric in (0,1). Significance level for forward inclusion. Default value is `0.05`.
p_backward	Numeric in (0,1). Significance level for backward elimination. Default value is `0.01`.
fit_function	Function. Fitting function that returns a fit-like object consumable by `get_ofv`. Default value is `sg_fit`.
update_theta_init	Logical. If `TRUE`, refreshes theta `INIT` values from accepted forward fits only (never from rejected candidates). Default value is `FALSE`.
run_backward	Logical. If `TRUE`, runs Stage 4 backward elimination after forward inclusion converges. Default value is `TRUE`.
update_theta_init_backward	Logical. If `TRUE`, refreshes theta `INIT` only after accepted backward removals. Default value is `FALSE`.
path_to_fitter	Character or `NULL`. Path to the fitter executable. When `NULL`, `gco$path_to_fitter` is used if present. Default value is `NULL`.

Value

A list with `final_gco`, `final_covariates`, forward/backward summaries, runtime `settings`, and execution `metadata`.

Examples

```
model_path <- tempfile(fileext = ".txt")
data_path <- tempfile(fileext = ".csv")
writeLines(c("[LONGITUDINAL]", "input = {CL, V}", "PK:"), model_path)
writeLines(c("ID,TIME,DV,WT", "1,0,0,70", "2,0,0,80"), data_path)

gco <- list(
  model = model_path,
  data = data_path,
```

```
  headers = list(
    list(name = "ID", use = "identifier", type = NULL),
    list(name = "TIME", use = "time", type = NULL),
    list(name = "DV", use = "observation", type = "continuous"),
    list(name = "WT", use = "covariate", type = "continuous")
  ),
  theta = data.frame(
    NAME = c("CL", "V"),
    TRANS = c("logNormal", "logNormal"),
    INIT = c(0.2, 20),
    EST = c(TRUE, TRUE),
    stringsAsFactors = FALSE
  ),
  ruv = list(dummy = TRUE),
  re = list(dummy = TRUE),
  occ = list(dummy = TRUE),
  covs = list(),
  project_name = "base_model"
)

gfo <- list(
  OFV = data.frame(LL = 100),
  SUMTAB = data.frame(PAR = character(0), VALUE = numeric(0), stringsAsFactors = FALSE),
  COTAB = data.frame(ID = 1:4, WT = c(70, 80, 90, 75), stringsAsFactors = FALSE),
  CATAB = data.frame(ID = 1:4, stringsAsFactors = FALSE)
)

mock_fit <- function(model, data, headers, theta, ruv, re, occ, covs, project_name,
                     task_opt = NULL, opt_name = "Monolix", fit = TRUE,
                     path_to_save_output = NULL, path_to_fitter = NULL) {
  ofv <- if (grepl("_001$", project_name)) 95 else 99
  list(
    GFO = list(
      OFV = data.frame(LL = ofv),
     SUMTAB = data.frame(PAR = character(0), VALUE = numeric(0), stringsAsFactors = FALSE),
      COTAB = data.frame(dummy = 1),
      CATAB = data.frame(dummy = 1)
    )
  )
}

result <- sg_covsearch(
  gfo = gfo,
  gco = gco,
  output_dir = tempfile("covsearch-example-"),
  covariates = "WT",
  parameters = "CL",
  run_backward = FALSE,
  fit_function = mock_fit
)

result$forward$selected[, c("parameter", "covariate")]
```

`sg_covsens_sim` *Covariate sensitivity analysis via simulation*

Description

Evaluate the impact of continuous and categorical covariates on model parameters and exposure metrics. For each covariate the function perturbs its value to selected quantiles (continuous) or observed categories (categorical) while holding the remaining covariates at their reference values, simulates from the pharmacometric model under parameter uncertainty, and summarises the resulting percent change relative to the reference.

Usage

```
sg_covsens_sim(
  fpath_i = NULL,
  ds_parest = NULL,
  ds_covs = NULL,
  model,
  stimes,
  et,
  est_covmat = NULL,
  npop = 5,
  cont_cov_l,
  cat_cov_l,
  quantiles = c(0.1, 0.9),
  outputs = "Cc",
  aggr = c("min", "max", "mean", "auc"),
  seed = NULL,
  ci = 95
)
```

Arguments

`fpath_i`	String, GFO, or a named list with a `GFO` component (and optionally `GCO`). If a string is given, the path to a `.RData` or `.json` file with a fit object is expected.
`ds_parest`	Data.frame. Parameter estimates table with columns `parameter` and `value`. Required when `fpath_i` is `NULL`; must be provided together with `ds_covs`. Default is `NULL`.
`ds_covs`	data.frame. Subject-level covariate dataset (one row per subject) containing both continuous and categorical covariate columns. Required when `fpath_i` is `NULL`; must be provided together with `ds_parest`. Default is `NULL`.
`model`	GMO. RxODE2 model to simulate from.
`stimes`	numeric vector. Sampling time points for steady-state simulation (e.g. generated by a cycling function over dosing intervals).

et — data.frame. Event (dosing) table. Must contain at least columns id, time, amt, evid and Regimen. Additional columns such as ii, addl, dur and Dose are used when present.

est_covmat — Data.frame. Parameter estimation covariance matrix. The first column (X1) must list parameter names; remaining columns (named identically) form the symmetric variance–covariance matrix.

npop — integer. Number of population-level simulation replicates drawn from the parameter uncertainty distribution. Default is 5.

cont_cov_l — A named list. Each element defines one continuous covariate and must itself be a list where par_vec is required; other components are optional/permissible. Name of each element should be equal the name of the continuous covariate:

- par_vec Character vector. Model parameter(s) affected by this covariate (e.g. c("CL")).
- UTNAME Character or NULL. Column name of the untransformed (back-transformed) covariate (e.g. "AGE"). If NULL or NA, defaults to name of the covariate. Default is NULL.
- REF Character or numeric. Reference value for the covariate. Use "median" to derive from data, or a numeric value. Default is "median".
- NICENAME Character or NULL. Display label for plots and tables (e.g. "Age, years"). Default is NULL.

cat_cov_l — A named list with. Each element defines one categorical covariate and must itself be a list where par_vec is required; other components are optional/permissible. Name of each element should be equal the name of the categorical covariate:

- par_vec Character vector. Model parameter(s) affected by this covariate (e.g. c("ka")).Required.
- NICENAME Character or NULL. Display label. Default is NULL.
- REF Character or NULL. Reference category value. If NULL, the first factor level (alphabetically) is used. Default is NULL.

quantiles — A numeric vector of length 2. Lower and upper quantiles of the continuous covariate distribution to test. Default is c(0.1, 0.9).

outputs — character vector. Name(s) of the model output variable(s) to evaluate (e.g. "Cc" or c("Cc", "Effect")). Default is "Cc".

aggr — character vector. Exposure aggregation metric(s) applied over the simulation time grid. Allowed values: "min" (Cmin), "max" (Cmax), "mean" (Cmean), "auc" (AUC).

seed — integer. Seed for the random number generator. Default is NULL.

ci — integer. Confidence interval (CI) level as a percentage. Allowed values: 95, 90, 80, 70, 50. Default is 95.

Details

Two data-source modes are supported (mutually exclusive):

- **File mode** — supply fpath_i (path to a saved object with GFO tables such as SUMTAB, CATAB, and COTAB).

- **Table mode** — supply both `ds_parest` (parameter estimates) and `ds_covs` (covariate dataset).

The analysis proceeds in two stages for each covariate type (continuous and categorical):

1. **Parameter sensitivity** — the covariate is set to its quantile or category value and the model is evaluated at time zero (no ODE integration) to quantify the direct effect on each parameter.
2. **Exposure sensitivity** — the full time-course is simulated over `stimes` and exposure metrics (`aggr`) are computed.

Uncertainty is propagated by sampling `npop` parameter vectors from a multivariate normal distribution parameterised by the population estimates and the estimation covariance matrix (`est_covmat`).

Value

A named list of four elements:

`$PARSENS` data.frame. Full sensitivity results for model parameters: percent change statistics (mean, median, percentiles) for each covariate–parameter combination, with columns `NICEN`, `VAR`, `KEY`, `LAB`, `Type` and summary statistics (`mean` through `P975`).

`$SUMPARSENS` data.frame. Compact summary table with columns `Parameter`, `Covariate`, `Cov. percentile`, `Cov. value`, `Mean` and `90%CI`.

`$EXPSENS` data.frame. Sensitivity of exposure metrics (Cmin, Cmax, Cmean, AUC) to covariates, structured identically to `$PARSENS`.

`$COVREF` data.frame. Reference values for each covariate used in the analysis, with columns `COV`, `NICEN`, `REF_VALUE` and `REF_SOURCE` (`"reference"` for user-specified values, `"median"` when `REF = "median"` was used for a continuous covariate).

See Also

`sg_sim`, `read_smrg_obj`

Examples

```
library(dplyr)
library(rxode2)

# --- Covariate definitions (only par_vec is required) ---
cont_cov_l <- list(
  LG_AGE = list(par_vec = c("CL")),
  LG_WEIGHT = list(
    UTNAME = "WEIGHT",
    NICENAME = "Weight, kg",
    par_vec = c("Vd")
  )
)

cat_cov_l <- list(
  SEX = list(par_vec = c("ka")),
  CYP2C9 = list(
    REF = "1",
    NICENAME = "CYP2C9 genotype",
```

```
    par_vec = c("CL")
  )
)

# --- Dosing ---
ev_t_input <- tribble(
  ~id, ~time, ~ii, ~amt, ~addl, ~dur, ~evid, ~Regimen,        ~Dose,
  1,   0,     336, 10,   21,    0.5,  1,     "0.3 mg/kg Q2W", 0.3
)
# --- Model ---
model <- RxODE({
  # Doses in mg
  # Time in hours

  ### Parameter values
  # Typical values
  ka_pop = 0.073;
  Vd_pop = 14.8;
  CL_pop = 0.347;

  # Random effects
  omega_ka = 0;
  omega_Vd = 0;
  omega_CL = 0;

  # Covariate effect
  # Continuous
  beta_CL_LG_AGE = 0.49990114;
  beta_Vd_LG_WEIGHT = 0.60529433;

  # Categorical
  beta_CL_CYP2C9_1_2 = -0.339;
  beta_CL_CYP2C9_1_3 = -0.574;
  beta_CL_CYP2C9_2_2 = -1.079;
  beta_CL_CYP2C9_2_3 = -0.745;
  beta_CL_CYP2C9_3_3 = -2.13;

  beta_ka_SEX_1 = -0.12198035;

  # Residual error
  Cc_b = 0;

  # Transformations
  ka_tv = exp(ka_pop);
  Vd_tv = exp(Vd_pop);
  CL_tv = exp(CL_pop);

  CL_multiplier = 1.0;  # Default/reference
  ka_multiplier = 1.0;

  if (SEX == "1") {ka_multiplier = exp(beta_ka_SEX_1)}

  if (CYP2C9 == "1") {
```

```
    CL_multiplier = exp(beta_CL_CYP2C9_1_2);
  } else if (CYP2C9 == "2") {
    CL_multiplier = exp(beta_CL_CYP2C9_1_3);
  } else if (CYP2C9 == "3") {
    CL_multiplier = exp(beta_CL_CYP2C9_2_2);
  } else if (CYP2C9 == "4") {
    CL_multiplier = exp(beta_CL_CYP2C9_2_3);
  } else if (CYP2C9 == "5") {
    CL_multiplier = exp(beta_CL_CYP2C9_3_3);
  }

  ka = ka_tv*ka_multiplier*exp(omega_ka);
  Vd = Vd_tv*exp(beta_Vd_LG_WEIGHT * LG_WEIGHT + omega_Vd); #Vd_tv*exp(omega_Vd);
  CL = CL_tv*CL_multiplier*exp(beta_CL_LG_AGE * LG_AGE + omega_CL);


  ### Explicit functions
  Cc = Ac/Vd;

  ### Initial conditions
  Ad(0) = 0;
  Ac(0) = 0;

  ### Differential equations
  d/dt(Ad) = - ka*Ad;
  d/dt(Ac) = ka*Ad - CL*Cc;

  Cc_ResErr = Cc*(1 + Cc_b);
})

# --- Estimation covariance (mock) ---
pnames     <- parest$parameter
npar       <- length(pnames)
set.seed(1)
m_cov      <- matrix(0.02, npar, npar)
diag(m_cov) <- 0.05 + runif(npar, 0, 0.05)
m_cov      <- (m_cov + t(m_cov)) / 2
est_covmat <- as_tibble(cbind(X1 = pnames, as.data.frame(m_cov)))
names(est_covmat)[-1] <- pnames

# --- Simulation times (steady-state cycle 10) ---
ss_cycle <- 10
stimes_ss <- c(
  ss_cycle * 4 * 7 * 24 + c(seq(0, 23.5, 0.5), seq(24, 335, 1)),
  ss_cycle * 4 * 7 * 24 + 2 * 7 * 24 + c(seq(0, 23.5, 0.5), seq(24, 335, 1))
)

# --- Run ---
result <- sg_covsens_sim(
  fpath_i=NULL, ds_parest = parest, ds_covs = ds_covval,
  model = model, stimes = stimes_ss, et = ev_t_input,
  est_covmat = est_covmat, npop = 10,
  cont_cov_l = cont_cov_l, cat_cov_l = cat_cov_l,
```

```
  quantiles = c(0.2, 0.8), aggr = c("max"),
  outputs = "Cc"
)
print(result[["PARSENS"]])
print(result[["SUMPARSENS"]])
print(result[["EXPSENS"]])
```

`sg_covsens_vis` *Visualisation of covariate sensitivity analysis results*

Description

Draw a forest-style graphic (points with uncertainty intervals) from the sensitivity tables produced by `sg_covsens_sim`. Each facet row corresponds to a model output or exposure metric (`VAR`); each point is a covariate scenario (`LAB`), coloured by covariate type (`Type`).

Usage

```
sg_covsens_vis(
  covsens_res,
  plot_type = c("PARSENS", "EXPSENS"),
  exclude_vars = NULL,
  ci = 95,
  ci_limits = c(0.8, 1.25),
  ci_band_alpha = 0.2,
  ci_band_col = "firebrick",
  ref_line_col = "grey25",
  palette = MSDcol[c(1, 3, 4, 5, 6, 7)],
  point_size = 2.5,
  errorbar_width = 0.2,
  lab_y = "standard",
  cap = "standard",
  var_nice_names = NULL
)
```

Arguments

`covsens_res`	Named list as returned by `sg_covsens_sim()`. Must contain the element selected by `type` (`PARSENS` and/or `EXPSENS` data.frames with columns `LAB`, `VAR`, `mean`, `Type`, and the interval columns named by `ci_quantiles`).
`plot_type`	Character scalar: `"PARSENS"` (default) or `"EXPSENS"`.
`exclude_vars`	Character vector. Contains `VAR` levels to omit (e.g. `"Cc_Cmin"`). `NULL` keeps all rows.
`ci`	Numeric. Confidence level in percents. Default is 95

ci_limits	Numeric vector of length 2: lower and upper bounds of the shaded acceptance band and of the dotted horizontal guides. Default c(0.8, 1.25) is a common bioequivalence-style window on the ratio scale.
ci_band_alpha	Numeric in $[0, 1]$: transparency of the shaded band. Default 0.2.
ci_band_col	Color for the band fill and dotted limit lines. Default "firebrick".
ref_line_col	Color for the dashed horizontal line at y = 1 (no change from reference). Default "grey25".
palette	Colors for the Type scale (continuous vs categorical covariates, etc.). Recycled if there are more levels than colors. Default MSDcol[c(1, 3, 4, 5, 6, 7)].
point_size	Point size for geom_point. Default 2.5.
errorbar_width	Numeric. Width argument for geom_errorbar. Default 0.2.
lab_y	Axis label for the numeric scale (this becomes the horizontal axis after coord_flip()). Default mentions a 95% interval; change if you use different ci_quantiles.
cap	Optional figure caption, passed to ggplot2::labs(caption = ...) (e.g. text describing reference covariate values). Default NULL.
var_nice_names	Named character vector. Display labels for exposure facet rows when plot_type = "EXPSENS". Names must match the unique values in VAR (after exclude_vars); length equals the number of those unique values. Default NULL (facets use VAR).

Details

Two views are available, matching the named elements of the simulation output:

- PARSENS — sensitivity of individual parameters (simulation at time zero, no ODE time course).
- EXPSENS — sensitivity of exposure summaries (e.g. Cmin, Cmax, Cavg) after full simulation over stimes in sg_covsens_sim.

Values on the y-axis are ratios relative to the reference scenario: 1 means no change. In sg_covsens_sim, percent change relative to reference is transformed to this scale before tabulation. The shaded region between ci_limits highlights a target interval; points whose intervals lie largely inside can be read as scenarios consistent with that criterion, subject to study-specific rules.

The plot layers are drawn in order: reference band, error bars, points, reference line at 1, dotted lines at ci_limits, then faceting by VAR and flipped coordinates so labels read along the vertical axis.

Value

A ggplot2 object (inactive until printed or saved). You can add further layers or themes with the usual **ggplot2** API.

See Also

sg_covsens_sim

Examples

```
library(tibble)
library(dplyr)
library(rxode2)
# Typical workflow: run the simulation (see examples in ?sg_covsens_sim),
# then visualise parameter and exposure sensitivity.

cont_cov_l <- list(
  LG_AGE = list(NAME = "LG_AGE", UTNAME = "AGE",
                REF = "median", NICENAME = "Age, years",
                par_vec = c("CL")),
  LG_WEIGHT = list(NAME = "LG_WEIGHT", UTNAME = "WEIGHT",
                REF = "median", NICENAME = "Weight, kg",
                par_vec = c("Vd"))
)

cat_cov_l <- list(
  SEX = list(NAME = "SEX", NICENAME = "Sex",
             REF = "0", par_vec = c("ka")),
  CYP2C9 = list(NAME = "CYP2C9", NICENAME = "CYP2C9 genotype",
                REF = NULL, par_vec = c("CL"))
)

# --- Dosing ---
ev_t_input <- tribble(
 ~id, ~time, ~ii, ~amt, ~addl, ~dur, ~evid, ~Regimen,        ~Dose,
  1,   0,     336, 10,   21,    0.5,  1,     "0.3 mg/kg Q2W", 0.3
)
# --- Model ---
model <- RxODE({
 # Doses in mg
  # Time in hours

 ### Parameter values
  # Typical values
 ka_pop = 0.073;
  Vd_pop = 14.8;
  CL_pop = 0.347;

  # Random effects
  omega_ka = 0;
  omega_Vd = 0;
  omega_CL = 0;

  # Covariate effect
  # Continuous
  beta_CL_LG_AGE = 0.49990114;
  beta_Vd_LG_WEIGHT = 0.60529433;

  # Categorical
  beta_CL_CYP2C9_1_2 = -0.339;
  beta_CL_CYP2C9_1_3 = -0.574;
```

```
  beta_CL_CYP2C9_2_2 = -1.079;
  beta_CL_CYP2C9_2_3 = -0.745;
  beta_CL_CYP2C9_3_3 = -2.13;

  beta_ka_SEX_1 = -0.12198035;

  # Residual error
  Cc_b = 0;

 # Transformations
  ka_tv = exp(ka_pop);
  Vd_tv = exp(Vd_pop);
  CL_tv = exp(CL_pop);

  CL_multiplier = 1.0;  # Default/reference
  ka_multiplier = 1.0;

  if (SEX == "1") {ka_multiplier = exp(beta_ka_SEX_1)}

  if (CYP2C9 == "1") {
    CL_multiplier = exp(beta_CL_CYP2C9_1_2);
  } else if (CYP2C9 == "2") {
    CL_multiplier = exp(beta_CL_CYP2C9_1_3);
  } else if (CYP2C9 == "3") {
    CL_multiplier = exp(beta_CL_CYP2C9_2_2);
  } else if (CYP2C9 == "4") {
    CL_multiplier = exp(beta_CL_CYP2C9_2_3);
  } else if (CYP2C9 == "5") {
    CL_multiplier = exp(beta_CL_CYP2C9_3_3);
  }

  ka = ka_tv*ka_multiplier*exp(omega_ka);
  Vd = Vd_tv*exp(beta_Vd_LG_WEIGHT * LG_WEIGHT + omega_Vd); #Vd_tv*exp(omega_Vd);
  CL = CL_tv*CL_multiplier*exp(beta_CL_LG_AGE * LG_AGE + omega_CL);


  ### Explicit functions
  Cc = Ac/Vd;

  ### Initial conditions
  Ad(0) = 0;
  Ac(0) = 0;

  ### Differential equations
  d/dt(Ad) = - ka*Ad;
  d/dt(Ac) = ka*Ad - CL*Cc;

  Cc_ResErr = Cc*(1 + Cc_b);
})

# --- Estimation covariance (mock) ---
pnames     <- parest$parameter
npar       <- length(pnames)
```

```
set.seed(1)
m_cov      <- matrix(0.02, npar, npar)
diag(m_cov) <- 0.05 + runif(npar, 0, 0.05)
m_cov      <- (m_cov + t(m_cov)) / 2
est_covmat <- as_tibble(cbind(X1 = pnames, as.data.frame(m_cov)))
names(est_covmat)[-1] <- pnames

# --- Simulation times (steady-state cycle 10) ---
ss_cycle <- 10
stimes_ss <- c(
  ss_cycle * 4 * 7 * 24 + c(seq(0, 23.5, 0.5), seq(24, 335, 1)),
  ss_cycle * 4 * 7 * 24 + 2 * 7 * 24 + c(seq(0, 23.5, 0.5), seq(24, 335, 1))
)
result <- sg_covsens_sim(
  fpath_i = NULL, ds_parest = parest, ds_covs = ds_covval,
  model = model, stimes = stimes_ss, et = ev_t_input,
  est_covmat = est_covmat, npop = 10,
  cont_cov_l = cont_cov_l, cat_cov_l = cat_cov_l,
  quantiles = c(0.1, 0.9), aggr = c("min", "max", "mean"),
  outputs = "Cc"
)

p_par <- sg_covsens_vis(result, plot_type = "PARSENS")
p_exp <- sg_covsens_vis(result, plot_type = "EXPSENS")
p_par
p_exp

# Alternate interval columns (must exist in the sensitivity tables)
p <- sg_covsens_vis(result, ci = 95)
p

# Drop selected metrics from the exposure panel
p <- sg_covsens_vis(result, plot_type = "EXPSENS", exclude_vars = c("Cc_Cmin"))
p
```

`sg_fit` *Optimize model with Monolix or Simurg core fitter*

Description

Optimize model with Monolix or Simurg core fitter

Usage

```
sg_fit(
  model,
  data,
  headers,
```

```
  theta,
  ruv,
  re,
  occ,
  covs,
  project_name,
  task_opt = NULL,
  opt_name = "Monolix",
  fit = FALSE,
  path_to_save_output = NULL,
  path_to_fitter = NULL,
  max_wait_time = 3600
)
```

Arguments

model	string. Path to a txt file file with model structure in Monolix syntax. This should be a valid file path pointing to a model file that defines the pharmacokinetic/pharmacodynamic model structure
data	String. Path to the dataset used to fit a model. Should be a CSV file containing the pharmacokinetic/pharmacodynamic data with appropriate column structure matching the headers specification
headers	List. A list specifying the data frame column headers. Each element should be a list containing column information with the following structure: • name - string, column name in the dataset • use - string, column usage type. Valid values include: – "identifier" for subject ID columns – "time" for time columns – "observation" for dependent variable columns – "observationtype" for observation type identifier – "administration" for administration route – "amount" for dose amount – "eventidentifier" for event ID – "missingdependentvariable" for missing DV flag – "covariate" for covariate columns • type - string or NULL, data type specification. For observations use "continuous", "count/categorical" or "event", depending on the nature of observations. For covariates use "continuous" or "categorical". Can be NULL for non-covariate columns.
theta	tibble or data.frame. Should contain the following columns: • NAME - string, parameter name. Should match names specified in the model file • TRANS - string, parameter transformation type. Can be: "normal", "logNormal", "logitNormal" • INIT - numeric, initial value for the parameter

- LB - numeric, lower bound for logit transformation, NA for other transformations
- UB - numeric, upper bound for logit transformation, NA for other transformation
- EST - logical, whether to estimate this parameter.

ruv — A list specifying options for residual error used in model fit, containing:

- YNAME - string, output name. Typically "y1", "y2",...
- DVID - numeric, observation type identifier corresponding to DVID column values
- TRANS - string, residual error distribution. Can be: "normal", "logNormal", "logitNormal"
- PRED - string, prediction variable name from the model
- ERR - string, error model type. Options include:
  - "constant" for additive error
  - "proportional" for proportional error
  - "combined1" for combined additive and proportional error
- INIT - numeric vector, initial values for error parameters (length depends on error model)
- EST - logical vector, whether to estimate each error parameter (same length as INIT)
- BLQM - below limit of quantification method (can be NULL) For several observation types list of lists should be provided: one residual error (ruv) object for each observations.

re — List. A list specifying options for random effects used in model fit, containing:

- init - matrix, initial values for the variance-covariance matrix of random effects. Rows and columns should correspond to parameters defined in theta. Diagonal elements represent variances, off-diagonal elements represent covariances. Use 0 for no variability
- est - matrix, logical matrix of same dimensions as init specifying which variance-covariance elements to estimate. Use TRUE to estimate, FALSE to fix, NA to not use this random effect.

occ — A list specifying interoccasion variability properties, containing:

- init - matrix, initial values for the interoccasion variance-covariance matrix. Same structure as re$init but for occasion-to-occasion variability. Use 0 for no interoccasion variability
- est - matrix, logical matrix specifying which interoccasion variance-covariance elements to estimate. Use TRUE to estimate, FALSE to fix, NA for elements not applicable. Typically all elements are NA when no interoccasion variability is modeled.

covs — A list of covariate structures. Each element should be a list containing covariate relationship definitions with the following structure:

- PAR - string, name of the parameter to which covariate effect is applied
- COVNAME - string, name of the covariate column in the dataset
- FUNC - string, functional form of the covariate effect

- TRANS - string, transformation applied to covariate
- REF - numeric, reference value for categorical covariates
- INIT - numeric, initial value for the covariate effect parameter
- EST - logical, whether to estimate the covariate effect Can be NULL, if no covariates are used. Default is NULL

project_name
: String. The name of the Monolix project without file extension. This will be used as the base name for output files and directories.

task_opt
: String. Additional task options to be passed to the fitting software. For Monolix, this can include specific task configurations or optimization settings. When NULL, default tasks (populationParameters, individualParameters, fim, logLikelihood) will be used. Default is NULL.

opt_name
: String. Specify the optimizer/fitter to use for model fitting. Currently supported options:

- "Monolix" - uses Monolix Suite for population pharmacokinetic modeling (generates .mlxtran files)
- "Simurg" - uses SimuRg internal fitter (generates .R files with JSON control structure) Default is "Monolix".

fit
: Logical. If TRUE, the model fitting will be executed immediately using the specified fitter. If FALSE, only the fit configuration file will be generated without running the fit. Set to FALSE for file preparation only, or TRUE to run the complete fitting process. Default is FALSE.

path_to_save_output
: String. Path to save fit output files. Should be a valid directory path where the fit results and project files will be saved. If NULL, current working directory will be used. Default is NULL.

path_to_fitter
: String. The path to the program fitter executable. For Monolix, this should point to the monolix.bat file. If NULL, "C:/ProgramData/Lixoft/MonolixSuite2023R1/bin/monolix.bat" will be used as default. Default is NULL.

max_wait_time
: numeric. Maximum time in seconds to wait for fit results to complete. Default is 3600 seconds (1 hour). Set to Inf for no timeout.

Value

If fit = TRUE, a named list with components GFO and GCO. If fit = FALSE, the fit configuration file is written and NULL is returned invisibly.

See Also

GCO, GFO, sg_converter()

Examples

```
library(tibble)
library(dplyr)
library(stringr)
# Specify structural model path. For fit, should be in Monolix syntax
model <- system.file("extdata", "models", "model_PK_1c.txt", package = "SimuRg")
```

```
# Specify data path. Should be ADPPK-like format
data  <- system.file("extdata", "datasets", "dspk-warf.csv", package = "SimuRg")

# Specify headers. Should be a list of lists with the following elements:
# name - string, name of the column
# use - string, use of the column from Monolix documentation
# type - string, type of the column. for use = "covariate", should be
# "continuous" or "categorical", for other uses should be NULL
 headers <- list(list(name = "ID", use = "identifier", type = NULL),
                 list(name = "TIME", use = "time", type = NULL),
                 list(name = "DV", use = "observation", type = "continuous"),
                 list(name = "DVID", use = "observationtype", type = NULL),
                 list(name = "ADM", use = "administration", type = NULL),
                 list(name = "AMT", use = "amount", type = NULL),
                 list(name = "EVID", use = "eventidentifier", type = NULL),
                 list(name = "MDV", use = "missingdependentvariable", type = NULL),
                 list(name = "AGE", use = "covariate", type = "continuous"),
                 list(name = "AGE_centered", use = "covariate", type = "continuous"),
                 list(name = "SEX", use = "covariate", type = "categorical"),
                 list(name = "WEIGHT", use = "covariate", type = "continuous"),
                 list(name = "BMI", use = "covariate", type = "continuous"))

# Dataset with the parameters properties. Should be a tibble with the following columns:
# NAME - string, name of the parameter
# TRANS - string, distribution of the parameter. Should be one of the following:
# "normal", "logNormal", "logitNormal"
# INIT - numeric, initial value of the parameter or its fixed value
# LB - numeric, lower bound of the parameter for logit transformation
# UB - numeric, upper bound of the parameter for logit transformation
# EST - logical, estimation status of the parameter. For estimation, should
# be TRUE, for fixed, should be FALSE
 theta <- tribble(~NAME, ~TRANS, ~INIT, ~LB, ~UB, ~EST,
                  "Cl", "logNormal", 0.2, NA, NA, TRUE,
                  "V", "logNormal", 20, NA, NA, TRUE,
                  "ka", "logNormal", 0.2, NA, NA, TRUE
 )
 # Examples of random effect model specification
 # Examples of random effect model specification for fit
 # Single observation (legacy format):
 # YNAME - string, name of the observation, ususally y1, y2, ...
 # DVID - numeric, observation type identifier corresponding to DVID column values
 # TRANS - string, residual error distribution. Can be: "normal", "logNormal",
 # "logitNormal"
 # PRED - string, prediction variable name from the model
 # ERR - string, error model type. Options include: "constant" for additive
 # error, "proportional" for proportional error, "combined1" for combined
 # additive and proportional error
 # INIT - numeric vector, initial values for error parameters
 # (length depends on error model)
 # EST - logical vector, whether to estimate each error parameter
 # (same length as INIT)
 # BLQM - below limit of quantification method (can be NULL)
 # Single observation (legacy format):
```

```
 ruv <- list(YNAME = "y1", DVID = 1, TRANS = "normal", PRED = "Cc",
             ERR = "combined1", INIT = c(1, 1), EST = c(TRUE, TRUE), BLQM = NULL)

 # Multiple observations (recommended format):
ruv <- list(
  list(YNAME = "y1", DVID = 1, TRANS = "normal", PRED = "Cc",
       ERR = "combined1", INIT = c(1, 1), EST = c(TRUE, TRUE), BLQM = NULL),
  list(YNAME = "y2", DVID = 2, TRANS = "normal", PRED = "EFF",
      ERR = "proportional", INIT = c(0.1), EST = c(TRUE), BLQM = NULL)
)

 # Example of random effects (RE) specification.
 #
 # init matrix: provides the initial values for the random effects.
 # est matrix: controls how each random effect is handled:
 #   TRUE  - the random effect will be estimated,
 #   FALSE - the random effect will be fixed at its initial value,
 #   NA    - no random effect will be applied.
 #
 # To fit a model without random effects, set all entries in the
 # est matrix to NA.
 #
 # The same logic applies to the between-occasion variability
 # matrix (occ).

 re <- list(init = tribble(~Cl, ~V, ~ka,
                           1, 0, 0,
                           0, 0, 0,
                           0, 0, 1) %>% as.matrix(),
            est = tribble(~Cl, ~V, ~ka,
                          TRUE, NA, NA,
                          NA, NA, NA,
                          NA, NA, TRUE) %>% as.matrix())
# Example of between-occasion variability (BOV) specification. The structure
# is the same as for RE, but for BOV

 occ <- list(init = tribble(~Cl, ~V, ~ka,
                            0, 0, 0,
                            0, 0, 0,
                            0, 0, 0) %>% as.matrix(),
             est = tribble(~Cl, ~V, ~ka,
                           NA, NA, NA,
                           NA, NA, NA,
                           NA, NA, NA) %>% as.matrix())
# Example of covariate specification. Should be a list of lists with the following elements:
# PAR - string, name of the parameter to which the covariate is applied
# COVNAME - string, name of the covariate
# FUNC - string, function to apply to the covariate. Should be "linear" for
# continuous covariates, "categorical" for categorical covariates
# TRANS - string, transformation of the covariate. Should be "median" for
# continuous covariates, "reference" for categorical covariates
# INIT - numeric, initial value of the covariate
# EST - logical, estimation status of the covariate. For estimation, should
```

```
#be TRUE, for fixed, should be FALSE
 covs <- list(list(PAR = "V", COVNAME = "AGE", FUNC = "linear",
                   TRANS = "median", INIT = 1, EST = TRUE),
              list(PAR = "ka", COVNAME = "SEX", REF = 0, INIT = 1, EST = TRUE))
 output_path <- str_c(tempdir(), "/")
 fitter_path <- "C:/ProgramData/Lixoft/MonolixSuite2023R1/bin/monolix.bat"
 # Examples of task_opt parameter

 task_opt <-  paste("populationParameters()", "individualParameters()",
                    "logLikelihood()", sep = "\n")
 task_opt_lin <-  paste("populationParameters()", "individualParameters()",
                        "fim(method = Linearization)",
                        "logLikelihood(method = Linearization)", sep = "\n")
# Generalized control object
# gco <-list(headers = headers,
#            data = data,
#            model = model,
#            task_opt = task_opt
#            covs = covs,
#            project_name = "test-proj",
#            theta = theta,
#            ruv = ruv,
#            re = re,
#            occ = occc,
#            modelText = "")
 result <- sg_fit(model, data, headers, theta, ruv, re, occ, covs,
                  project_name = "my_project", fit = FALSE, # set fit = TRUE for fit
                  path_to_save_output =  output_path,
                  path_to_fitter = fitter_path)
```

`sg_globalsens_sim` *Global sensitivity analysis via PRCC or eFAST*

Description

Performs global sensitivity analysis using either PRCC (Partial Rank Correlation Coefficients with LHS sampling) or eFAST (extended Fourier Amplitude Sensitivity Test). For each sampled parameter set, the function runs simulations via `sg_sim()`, reduces trajectories to scalar GSO summaries, and computes sensitivity indices for each output–statistic pair.

Usage

```
sg_globalsens_sim(
  method = c("PRCC", "eFAST"),
  model,
  params,
  par_bounds,
  n_sim,
```

```
  stimes,
  output,
  stat_comp = c("mean", "min", "max"),
  et,
  theta = NULL,
  cov = NULL
)
```

Arguments

method	Character string. "PRCC" or "eFAST".
model	GMO. RxODE2 model to simulate from.
params	Character vector of parameter names to vary.
par_bounds	A tibble or data frame with columns PAR, LB, UB.
n_sim	Integer. Number of samples (LHS size for PRCC, base frequency size for eFAST).
stimes	Numeric vector. Sampling time points; a component of GSI. Default is NULL.
output	A character vector of outputs to keep. Passed to sg_sim().
stat_comp	A character vector of summary statistics to compute. Supported internally: "mean","median","min","max","sd","cmax","SS".
et	Data.frame. Event table; a component of GSI.
theta	Named vector or data frame. Population or baseline parameter values; a component of GSI. For sensitivity analysis, parameters listed in params are replaced by sampled values. Default is NULL.
cov	Covariates passed to sg_sim(). Default is NULL.

Value

List with:

result Tibble containing sensitivity analysis results. For PRCC method columns include: PAR, VAR, STAT, estimate, p.value. For eFAST method columns include: PAR, VAR, STAT, TYPE, VALUE, METHOD.

summary	Tibble of scalar summaries used for sensitivity computation: sim.id, VAR, STAT, value.
design	Data.frame of sampled parameter values (real scale) with sim.id. For PRCC this corresponds to LHS samples. For eFAST this corresponds to the FAST design matrix.
bounds	Parameter bounds used in the analysis.

Examples

```
# Example model
library(rxode2)
library(tibble)
```

```
source(system.file("extdata", "RxODE_model", "example_rxode_model.R", package = "SimuRg")) # mod_ex


# Set up event table
et_base <- tribble(
  ~id, ~time, ~evid, ~cmt, ~amt, ~addl, ~ii, ~IGFR, ~POPN,
  1,   0,     1,     1,    10,   2,     24,  112,   1
)

# Define parameter bounds
inits <- rxInits(mod_ex)
par_bounds <- tibble::tibble(
  PAR = c("POPCL", "POPVC"),
  LB  = inits[c("POPCL", "POPVC")] * (1 - 0.9),
  UB  = inits[c("POPCL", "POPVC")] * (1 + 0.9)
)

# Run eFAST sensitivity analysis
res_efast <- sg_globalsens_sim(
  method = c("eFAST"),
  model = mod_ex,
  params = c("POPCL"),
  par_bounds = par_bounds,
  n_sim = 100,
  stimes = seq(0, 168, 10),
  output = "Cc",
  cov = c("IGFR", "POPN"),
  et = et_base,
  stat_comp = c("mean")
)

# Run PRCC sensitivity analysis
res_prcc <- sg_globalsens_sim(
  method = c("PRCC"),
  model = mod_ex,
  params = c("POPCL", "POPVC"),
  par_bounds = par_bounds,
  n_sim = 100,
  stimes = seq(0, 168, 10),
  output = "Cc",
  cov = c("IGFR", "POPN"),
  et = et_base,
  stat_comp = c("mean")
)
```

`sg_globalsens_vis` *Visualize global sensitivity analysis results*

Description

Generates visualization plots for results produced by `sg_globalsens_sim()`. Supported visualizations:

- PRCC: heatmap or barplot
- eFAST: barplot of first and total order indices

Usage

```
sg_globalsens_vis(
  x,
  type = c("heatmap", "bar"),
  params = NULL,
  vars = NULL,
  stats = NULL
)
```

Arguments

`x`	Result object returned by `sg_globalsens_sim()`
`type`	Plot type for PRCC: `"heatmap"` or `"bar"`
`params`	Optional vector of parameters to display
`vars`	Optional vector of output variables
`stats`	Optional vector of statistics to display

Value

ggplot object

Examples

```
# Example model
library(rxode2)
library(tibble)
source(system.file("extdata", "RxODE_model", "example_rxode_model.R", package = "SimuRg")) # mod_ex


# Set up event table
et_base <- tribble(
  ~id, ~time, ~evid, ~cmt, ~amt, ~addl, ~ii, ~IGFR, ~POPN,
  1,   0,     1,     1,    10,   2,     24,  112,   1
)

# Define parameter bounds
inits <- rxInits(mod_ex)
par_bounds <- tibble::tibble(
  PAR = c("POPCL", "POPVC"),
  LB  = inits[c("POPCL", "POPVC")] * (1 - 0.9),
  UB  = inits[c("POPCL", "POPVC")] * (1 + 0.9)
```

```
)

# Run eFAST sensitivity analysis
res_efast <- sg_globalsens_sim(
  method = c("eFAST"),
  model = mod_ex,
  params = c("POPCL"),
  par_bounds = par_bounds,
  n_sim = 100,
  stimes = seq(0, 168, 10),
  output = "Cc",
  cov = c("IGFR", "POPN"),
  et = et_base,
  stat_comp = c("mean")
)

  efast_p <- sg_globalsens_vis(
    res_efast,
    params = c("POPCL"),
    vars = "Cc",
    stats = c("mean")
  )

  efast_p
# Run PRCC sensitivity analysis
res_prcc <- sg_globalsens_sim(
  method = c("PRCC"),
  model = mod_ex,
  params = c("POPCL", "POPVC"),
  par_bounds = par_bounds,
  n_sim = 100,
  stimes = seq(0, 168, 10),
  output = "Cc",
  cov = c("IGFR", "POPN"),
  et = et_base,
  stat_comp = c("mean")
)

  prcc_p <- sg_globalsens_vis(
    res_prcc,
    type = c("heatmap"),
    params = c("POPCL", "POPVC"),
    vars = "Cc",
    stats = c("mean")
  )

  prcc_p
```

`sg_gof_obpr` *Observed vs predicted plot*

Description

Function generates observed versus predicted scatter plots from GFO `$SDTAB`, a fundamental goodness-of-fit diagnostic tool in pharmacometric modeling. This visualization assesses model adequacy by comparing observed clinical measurements against model-predicted values, enabling identification of systematic bias, heteroscedasticity, and model misspecification patterns. Function contains options for faceting, coloring by covariates, and trend lines.

Usage

```
sg_gof_obpr(
  fpath_i,
  DVID = 1,
  cov_cols = NULL,
  indiv = TRUE,
  addline = TRUE,
  alpha_i = 0.5,
  smooth = TRUE,
  log_axes = FALSE,
  sc_factor = 1,
  abreaks = scales::pretty_breaks(7),
  lab_x = "Model-predicted values",
  lab_y = "Observed values",
  col_i = NULL,
  col_lab = NULL,
  facet_i = NULL,
  f_scales = "fixed",
  no_leg = FALSE,
  n_quantiles = 3,
  levels_discrete = 10
)
```

Arguments

`fpath_i`	String, GFO, or a named list with a `GFO` component (and optionally `GCO`). If a string is given, the path to a `.RData` or `.json` file with a fit object is expected.
`DVID`	Restrict `SDTAB` to one observation type. Numeric values select the `DVID` column (default `1`). If `SDTAB` has `DVNAME`, a character or factor is matched to `DVNAME` first; otherwise a digit-only string is coerced to numeric `DVID`.
`cov_cols`	A character vector specifying the names of the columns with covariates.
`indiv`	Logical. If `TRUE`, use individual predictions (`"IPRED"`); otherwise use population predictions (`"PRED"`). Default is `TRUE`.
`addline`	Logical. If `TRUE`, lines connecting observations of individual subjects will be added. Default is `TRUE`.

alpha_i	Numeric. Transparency level (from 0 to 1) for points/lines. Default is 0.5.
smooth	Logical. Add LOESS smooth line. Default is TRUE.
log_axes	Logical. If TRUE, a logarithmic scale is applied to all axes. Default is FALSE.
sc_factor	Numeric. Scaling factor for DV/PRED/IPRED values. Default is 1 (no scaling).
abreaks	A function that generates axis breaks. Default is scales::pretty_breaks(7).
lab_x	X-axis label. Default "Model-predicted values"
lab_y	Y-axis label. Default "Observed values"
col_i	String. Column name for color
col_lab	String. Label for color legend
facet_i	Character vector. Column name(s) for facet panels. Multiple columns are combined with + in facet_wrap. Default is NULL.
f_scales	String, one of "fixed", "free", "free_x", "free_y". User can specify whether the scales (x and y axes) should be fixed across all panels ("fixed"), free for each panel ("free"), or free only in one dimension ("free_x" or "free_y"). Default is "fixed"
no_leg	Logical. If TRUE, the legend is not shown. Default is FALSE
n_quantiles	Integer. Number of quantile groups for continuous variables in col_i. Default is 3.
levels_discrete	Integer. Maximum unique values to consider a variable discrete. Default is 10.

Value

A ggplot2 object

Examples

```
# Basic example with mock data
set.seed(123)  # For reproducibility
mock_obj <- list(
 SDTAB = data.frame(
   ID = rep(1:3, each = 5),
   TIME = rep(c(0, 1, 2, 4, 8), 3),
   DV = rnorm(15, mean = 10, sd = 2),
   PRED = rnorm(15, mean = 10, sd = 1.5),
   IPRED = rnorm(15, mean = 10, sd = 1.2),
   MDV = rep(0, 15)
 ),
 COTAB = data.frame(ID = 1:3, AGE = c(30, 45, 60)),
 CATAB = data.frame(ID = 1:3, RACE = c("Hispanic", "Hispanic", "Asian"))
)

# Basic plot
p <- sg_gof_obpr(mock_obj)
p

# With covariates and faceting
```

```
p <- sg_gof_obpr(
  mock_obj,
  cov_cols = "RACE",
  col_i = "RACE",
  facet_i = "RACE"
)
p
```

`sg_gof_par_cov` *Plot random effects or individual parameters vs covariates*

Description

Generates ggplot2 visualizations for either random effects (RE) or individual parameters (IndPar) versus continuous or categorical covariates from a GFO.

Usage

```
sg_gof_par_cov(fpath_i, ptype = "REvsCov", cat_cov = NULL, cont_cov = NULL)
```

Arguments

`fpath_i`	String, GFO, or a named list with a `GFO` component (and optionally `GCO`). If a string is given, the path to a `.RData` or `.json` file with a fit object is expected.
`ptype`	Character. Type of plot: `"REvsCov"` for random effects or `"IndParvsCov"` for individual parameters.
`cat_cov`	Optional tibble with categorical covariates. Must have columns `COV` and optionally `COVNAME` for labels.
`cont_cov`	Optional tibble with continuous covariates. Must have columns `COV` and optionally `COVNAME` for labels.

Value

A list of ggplot objects. Returns versus continious covariates `vs_contcov` and versus categorical covariates `vs_catcov` plots.

Examples

```
library(tibble)
cont_cov <- tibble(
  COV = c("AGE", "WEIGHT"),
  COVNAME = c("Age, years", "Body weight, kg")
)
cat_cov <- tibble(
  COV = c("SEX", "VKORC1_gentyp"),
  COVNAME = c("Sex, M/F", "VKORC1 genotype")
```

```
)
fpath_i <- system.file("extdata", "simurg_object", "Warfarin_PK.RData", package = "SimuRg")
p <- sg_gof_par_cov(
  fpath_i = fpath_i,
  ptype = "IndParvsCov",
  cont_cov = cont_cov,
  cat_cov = cat_cov
)
p$vs_contcov
p$vs_catcov
```

`sg_gof_par_dist` *Plot distributions of individual parameters with optional theoretical density*

Description

This function creates histograms of individual parameter estimates, with optional overlay of the theoretical distribution based on the population parameters and OMEGA matrix.

Usage

```
sg_gof_par_dist(
  fpath_i,
  par_seq = NULL,
  par_type = "Ind",
  n_bins = 30,
  tdist = TRUE,
  plot_type = "DIST"
)
```

Arguments

`fpath_i`	String, GFO, or a named list with a `GFO` component (and optionally `GCO`). If a string is given, the path to a `.RData` or `.json` file with a fit object is expected.
`par_seq`	Vector of strings. Character vector of parameter names to be plotted. If `NULL`, all parameters be included. Default is `NULL`
`par_type`	String. A character string specifying the type of parameters used for theoretical distribution overlay (only relevant when `plot_type = 'DIST'` and `tdist = TRUE`). If 'Ind' - Individual (default), distributions are shown on the natural parameter scale assuming log-normal variability. If 'RE' - Random Effect, distributions are shown on the ETA scale, assuming a normal distribution with mean zero and covariance defined by `$OMEGAMAT`, without transformation.
`n_bins`	Integer. Number of bins to use in the histogram. Default is 30.

tdist	Logical. If TRUE, overlay theoretical parameter distributions based on population mean and OMEGA matrix. Default is TRUE.
plot_type	Character string specifying the type of plot to generate. Options: 'DIST' for parameter distributions (default), 'QQ' for Q-Q plots, 'correlations' for correlation matrix plot.

Details

The function visualizes the distribution of post hoc individual parameter estimates (from GFO $PATAB) and optionally compares them with the expected population-level variability using $SUMTAB$DISTRIBUTION (see Details). Shrinkage values are calculated based on ETAshrinkage_var from $SUMTAB and included in facet labels.

Theoretical distributions are generated by sampling from a multivariate normal distribution (mean zero, covariance from $OMEGAMAT) and mapping to the individual-parameter scale using typical values and the DISTRIBUTION column: logNormal uses TV * exp(ETA), normal uses TV + ETA, logitNormal uses the inverse logit of logit(TV) + ETA. Parameters with missing or empty DISTRIBUTION do not get a theoretical curve.

Value

A ggplot object containing histogram(s) of individual parameter distributions, optionally overlaid with theoretical densities.

Examples

```
# Basic usage: distribution plots for all parameters
fpath_i <- system.file("extdata", "simurg_object", "Warfarin_PK.RData", package = "SimuRg")
p <- sg_gof_par_dist(fpath_i = fpath_i)
p

# Distribution plots for selected parameters (natural scale)
p <- sg_gof_par_dist(fpath_i = fpath_i, par_seq = c("ka", "Cl"))
p

# Distribution plots with theoretical densities disabled
p <- sg_gof_par_dist(fpath_i = fpath_i, tdist = FALSE)
p

# Distribution plots for ETA
p <- sg_gof_par_dist(fpath_i = fpath_i, par_seq = c("eta_ka", "eta_Cl"), par_type = "RE")
p

# Q-Q plots for all ETA parameters
p <- sg_gof_par_dist(fpath_i = fpath_i, plot_type = "QQ")
p

# Q-Q plot for a specific ETA parameter
p <- sg_gof_par_dist(fpath_i = fpath_i, plot_type = "QQ", par_seq = "eta_ka")
p

# Correlation matrix for selected parameters
```

```
p <- sg_gof_par_dist(fpath_i = fpath_i, plot_type = "correlations")
p
```

`sg_gof_res` *Create residual diagnostic plots*

Description

Generates residual diagnostic plots versus time/predictions from GFO `$SDTAB`. Supports faceting, covariate coloring, quantile binning for continuous covariates, and optional smoothing.

Usage

```
sg_gof_res(
  fpath_i,
  DVID = 1,
  cov_cols = NULL,
  indiv = TRUE,
  vs_time = TRUE,
  weighted = TRUE,
  addline = TRUE,
  alpha_i = 0.5,
  smooth = TRUE,
  log_x = FALSE,
  abreaks = scales::pretty_breaks(7),
  lab_y = NULL,
  lab_x = NULL,
  col_i = NULL,
  col_lab = NULL,
  facet_i = NULL,
  f_scales = "fixed",
  n_bins = 50,
  min_y = NA,
  max_y = NA,
  min_x = NA,
  max_x = NA,
  legend_fl = FALSE,
  n_quantiles = 3,
  levels_discrete = 10
)
```

Arguments

`fpath_i`	String, GFO, or a named list with a `GFO` component (and optionally `GCO`). If a string is given, the path to a `.RData` or `.json` file with a fit object is expected.

DVID	Restrict SDTAB to one observation type. Numeric values select the DVID column (default 1). If SDTAB has DVNAME, a character or factor is matched to DVNAME first; otherwise a digit-only string is coerced to numeric DVID.
cov_cols	A character vector specifying the names of the columns with covariates.
indiv	Logical. If TRUE, use individual predictions ("IPRED"); otherwise use population predictions ("PRED"). Default is TRUE.
vs_time	Logical. If TRUE, plot residuals vs TIME; otherwise, plot residuals vs predictions (IPRED/PRED).
weighted	Logical. If TRUE, use weighted residuals; otherwise, use RES/IRES
addline	Logical. If TRUE, lines connecting observations of individual subjects will be added. Default is TRUE.
alpha_i	Numeric. Transparency level (from 0 to 1) for points/lines. Default is 0.5.
smooth	Logical. Add LOESS smooth line. Default is TRUE.
log_x	Logical. If TRUE, a logarithmic scale is applied to x-axis. Default is FALSE.
abreaks	A function that generates axis breaks. Default is scales::pretty_breaks(7).
lab_y	String. Y-axis label.
lab_x	String. X-axis label.
col_i	String. Column name for color
col_lab	String. Label for color legend
facet_i	Character vector. Column name(s) for facet panels. Multiple columns are combined with + in facet_wrap. Default is NULL.
f_scales	String, one of "fixed", "free", "free_x", "free_y". User can specify whether the scales (x and y axes) should be fixed across all panels ("fixed"), free for each panel ("free"), or free only in one dimension ("free_x" or "free_y"). Default is "fixed"
n_bins	Integer. Number of bins to use in the histogram. Default is 30.
min_y	Numeric. Y-axis minimum limit. Default is NA.
max_y	Numeric. Y-axis maximum limit. Default is NA.
min_x	Numeric. X-axis minimum limit. Default is NA.
max_x	Numeric. X-axis maximum limit. Default is NA.
legend_fl	Logical. Show legend. Default is FALSE.
n_quantiles	Integer. Number of quantile groups for continuous variables in col_i. Default is 3.
levels_discrete	Integer. Maximum unique values to consider a variable discrete. Default is 10.

Value

A ggplot2 object

Examples

```
# Basic example with mock data
set.seed(123)  # For reproducibility
n_subjects <- 50
mock_obj <- list(
  SDTAB = do.call(rbind, lapply(1:n_subjects, function(id) {
    n_obs <- 6
    times <- sort(runif(n_obs, min = 0, max = 24))  # random times between 0 and 24h
    data.frame(
      ID    = id,
      TIME  = times,
      DV    = rnorm(n_obs, mean = 10, sd = 2),
      PRED  = rnorm(n_obs, mean = 10, sd = 1.5),
      IPRED = rnorm(n_obs, mean = 10, sd = 1.2),
      IWRES = rnorm(n_obs, mean = 0, sd = 0.8),
      IRES  = rnorm(n_obs, mean = 0, sd = 1.2),
      MDV   = 0
    )
  })),
  COTAB = data.frame(
    ID  = 1:n_subjects,
    AGE = sample(20:80, n_subjects, replace = TRUE)
  ),
  CATAB = data.frame(
    ID   = 1:n_subjects,
    RACE = sample(c("Hispanic", "Asian", "Caucasian"), n_subjects, replace = TRUE)
  )
)

# Basic plot: individual weighted residuals vs TIME (weighted = TRUE, vs_time = TRUE)
p <- sg_gof_res(mock_obj, smooth = FALSE)
p

# With covariates and faceting (use RACE as facet and AGE as color)
p <- sg_gof_res(
  mock_obj,
  smooth = TRUE,
  cov_cols = c("RACE","AGE"),
  col_i = "RACE",
  facet_i = "AGE",
  indiv = TRUE,
  weighted = TRUE,
  vs_time = FALSE,
  legend_fl = TRUE
)
p
```

---

sg_gof_res_dist *Plot distribution or QQ-plot of residuals from NONMEM/Simurg run*

Description

This function visualizes residuals from GFO `$SDTAB`. The residuals can be plotted either as histograms with overlaid normal density curves, or as QQ-plots to assess normality.

Usage

```
sg_gof_res_dist(
  fpath_i,
  res_type = "RES",
  DVID = 1,
  n_bins = 30,
  ndist = TRUE,
  plot_type = "DIST"
)
```

Arguments

`fpath_i`	String, GFO, or a named list with a `GFO` component (and optionally `GCO`). If a string is given, the path to a `.RData` or `.json` file with a fit object is expected.
`res_type`	Character vector. One or several types of residuals to plot. Values must correspond to column name(s) in `smrg_obj$SDTAB`, e.g. `"RES"`, `"IWRES"`, `"IRES"`. If multiple residual types are provided, plots will be generated for each of them.
`DVID`	Restrict `SDTAB` to one observation type. Numeric values select the `DVID` column (default `1`). If `SDTAB` has `DVNAME`, a character or factor is matched to `DVNAME` first; otherwise a digit-only string is coerced to numeric `DVID`.
`n_bins`	Integer. Number of bins to use in the histogram. Default is 30.
`ndist`	Logical. If `TRUE`, overlays the corresponding normal density curve on the histogram (default: TRUE).
`plot_type`	Character. Type of plot to produce: • `"DIST"` (default) - histogram of individual parameters, • `"QQ"` - QQ-plot of individual parameters.

Value

A `ggplot` object.

Examples

```
# Plot the distribution of individual weighted residuals (IWRES)
# (default \code{DVID = 1}; pass \code{DVID} or a \code{DVNAME} string to select another endpoint)
fpath_i <- system.file("extdata", "simurg_object", "Warfarin_PK.RData",
                       package = "SimuRg")
p <- sg_gof_res_dist(fpath_i = fpath_i, res_type = "IWRES")
p

# QQ-plots for the same default endpoint
p <- sg_gof_res_dist(fpath_i = fpath_i, res_type = "RES", plot_type = "QQ")
p
```

```
# Several residual columns at once (still one \code{DVID} unless you change it)
p <- sg_gof_res_dist(fpath_i = fpath_i, res_type = c("IWRES", "IRES"))
p
```

---

sg_gof_tp *Plot time profiles of observed and predicted values from GFO* $SDTAB.

Description

Plot time profiles of observed and predicted values from GFO $SDTAB.

Usage

```
sg_gof_tp(
  fpath_i,
  pop = TRUE,
  filt = "T",
  cov_cols = NULL,
  col_i = NULL,
  DVID = 1,
  tsld = FALSE,
  f_scales = "free",
  sort_by = NULL,
  desc = FALSE,
  log_y = FALSE,
  lab_x = "Time since first dose, h",
  lab_y = "Plasma concentration, mmol/L",
  cap = str_c("empty circles - observed data\n", "solid lines - ",
   "individual predictions\n", "dashed grey lines ", "- population predictions"),
  n_quantiles = 3,
  levels_discrete = 10
)
```

Arguments

fpath_i	String, GFO, or a named list with a GFO component (and optionally GCO). If a string is given, the path to a .RData or .json file with a fit object is expected.
pop	Logical. If TRUE (default), adds population predictions (PRED) as dashed lines and points.
filt	String. Provide a filter to apply. Default is "T"
cov_cols	A character vector specifying the names of the columns with covariates.
col_i	String. Column name for color
DVID	Restrict SDTAB to one observation type. Numeric values select the DVID column (default 1). If SDTAB has DVNAME, a character or factor is matched to DVNAME first; otherwise a digit-only string is coerced to numeric DVID.

tsld	Logical. If TRUE, uses time since last dose instead of time from first dose. Default is FALSE.
f_scales	String, one of "fixed", "free", "free_x", "free_y". User can specify whether the scales (x and y axes) should be fixed across all panels ("fixed"), free for each panel ("free"), or free only in one dimension ("free_x" or "free_y"). Default is "fixed"
sort_by	Character vector of column names to sort ID facets by. Use "DOSE" to sort by last dose. Other names (e.g. covariates) must exist in GFO.
desc	Logical. If TRUE, sort in descending order for all columns in sort_by.
log_y	Logical. If TRUE, a logarithmic scale is applied to y-axis. Default is FALSE.
lab_x	String. X-ax label. Default is "Time since first dose, h"
lab_y	String. Y-ax label. Default is "Plasma concentration, mmol/L"
cap	String. Plot caption. Default is "empty circles - observed data solid lines - individual predictions dashed grey lines - population predictions"
n_quantiles	Integer. Number of quantile groups for continuous variables in col_i. Default is 3.
levels_discrete	Integer. Maximum unique values to consider a variable discrete. Default is 10.

Value

A list of plots with predicted time profiles, faceted by id

Examples

```
fpath_i <- system.file("extdata", "simurg_object", "Warfarin_PK.RData",
                       package = "SimuRg")
plot_list <- sg_gof_tp(fpath_i)
plot_list[[1]]
```

---

sg_localsens_sim *Perform local sensitivity analysis simulations*

Description

Generates simulation datasets for a model varying each specified parameter within provided bounds or relative percentage. Supports multiple outputs and covariates.

Usage

```
sg_localsens_sim(
  model,
  params,
  stimes,
  output = NULL,
  perc = 0.2,
  lb = NULL,
  ub = NULL,
  cov = NULL,
  et,
  theta = NULL,
  n_sim = 10
)
```

Arguments

`model`	GMO. RxODE2 model object.
`params`	Character vector of parameter names to vary in the analysis.
`stimes`	Numeric vector of time points for the simulation.
`output`	Character vector of model outputs (variables) to keep. Default is NULL (all outputs).
`perc`	Numeric. If lb/ub not provided, defines ±percentage range around parameter base value. Default 0.2.
`lb`	Numeric vector of lower bounds for each parameter. Length must match `params`. Default NULL.
`ub`	Numeric vector of upper bounds for each parameter. Length must match `params`. Default NULL.
`cov`	Character vector of covariate names to include in the simulation. Default NULL.
`et`	Event table for the simulation.
`theta`	Named numeric vector of baseline parameters. Default NULL.
`n_sim`	Integer. Number of points to simulate between LB and UB for each parameter. Default 10.

Value

A tibble containing the simulated results with the following columns:

NPOP Optional. Population identification number. From simulation.

ID Optional. Individual subject ID. From simulation.

TIME Time points of simulation.

VAR Name of the output variable (model compartment or biomarker).

VALUE Simulated value of the variable.

mean, median, min, max, sd, etc. Optional. Aggregated statistics, if aggregation applied.

COV1,...COVn Optional. Covariate values, if cov provided.

PARNAME Name of the parameter being varied in this simulation.

PARVAL Value of the parameter used in this simulation.

PARVAL_NORM Normalized parameter value between 0 and 1 relative to LB and UB. Useful for plotting color gradients.

Examples

```
library(rxode2)
library(tibble)
mod_ex <- RxODE({
  # Doses in mg
  # Time in hours
  ### Parameter values
  # Typical
  POPCL = 5;
  POPVC = 180;
  POPQ = 7;
  POPVP = 52;
  POPKTR = 6;
  FBIOPAR = 1;
  # Covariate coefficients
  POPIGFRCOV = 0.8;
  POPPATCOVCLGR1 = 0.85;
  POPPATCOVCLGR2 = 0.85;
  POPPATCOVCLGR3 = 0.85;
  POPPATCOVCLGR4 = 0.85;
  # Random effects
  PPVCL = 0;
  PPVVC = 0;
  PPVKTR = 0;
  # Residual error
  RUV = 0;
  ### Covariates
  PATCOVCL = 1
  if(POPN == 1){PATCOVCL = POPPATCOVCLGR1}
  if(POPN == 2){PATCOVCL = POPPATCOVCLGR2}
  if(POPN == 3){PATCOVCL = POPPATCOVCLGR3}
  if(POPN == 4){PATCOVCL = POPPATCOVCLGR4}
  IGFRCOV = (IGFR/112)^POPIGFRCOV
  ## Parameters
  CL = POPCL * IGFRCOV * PATCOVCL * exp(PPVCL);
  VC = POPVC * exp(PPVVC);
  Q = POPQ;
  VP = POPVP;
  KTR = POPKTR * exp(PPVKTR);
  ### Explicit functions
  Cc = Ac/VC;                 # nmol/L
  Cp = Ap/VP;                 # nmol/L
  ### Initial conditions
  At1(0) = 0;         # mg
  At2(0) = 0;         # mg
```

```
  At3(0) = 0;         # mg
  At4(0) = 0;         # mg
  At5(0) = 0;         # mg
  At6(0) = 0;         # mg
  Ad(0) = 0;          # mg
  Ac(0) = 0;          # mg
  Ap(0) = 0;          # mg
  ### ODEs
  d/dt(At1) = - KTR*At1;
  d/dt(At2) = KTR*At1 - KTR*At2;
  d/dt(At3) = KTR*At2 - KTR*At3;
  d/dt(At4) = KTR*At3 - KTR*At4;
  d/dt(At5) = KTR*At4 - KTR*At5;
  d/dt(At6) = KTR*At5 - KTR*At6;
  d/dt(Ad) = KTR*At6 - KTR*Ad;
  d/dt(Ac) = KTR*Ad - CL*Cc - Q*(Cc - Cp);
  d/dt(Ap) = Q*(Cc - Cp);
  FBIO = FBIOPAR
  f(At1) = FBIO*1000000/505;
  CHECKRUV = RUV;
  Cc_ResErr = Cc + RUV*Cc;
})
et_base <- tribble(
  ~id, ~time, ~evid, ~cmt, ~amt, ~addl, ~ii, ~IGFR, ~POPN,
  1,   0,     1,     1,    10,   2,     24,  112,   1
)
sens_loc <- sg_localsens_sim(
  model = mod_ex,
  params = c("POPCL", "POPVC"),
  stimes = seq(0, 168, 0.1),
  output = "Cc",
  perc = 0.9,
  cov = c("IGFR", "POPN"),
  et = et_base
)
```

`sg_localsens_vis` *Visualize Local Sensitivity Analysis Results*

Description

This function creates plots to visualize the results of a local sensitivity analysis: (1) a family of curves plot showing how model outputs vary over time for different parameter perturbations, and (2) a tornado plot summarizing the relative influence of each parameter on selected summary metrics (e.g., output value at a chosen time or Cmax).

Usage

```
sg_localsens_vis(
  sens_data,
  tornado_time = NULL,
  metrics = "value",
  ref_data = NULL,
  log_scale = FALSE,
  color_low = "#1f77b4",
  color_high = "#ff7f0e",
  facet_scales = "free"
)
```

Arguments

sens_data	A data frame containing sensitivity analysis results from sg_localsens_sim()
tornado_time	Numeric value specifying the time point (in the same units as TIME) at which to evaluate the model output for the tornado plot. Defaults to the maximum time in sens_data if not specified.
metrics	Character vector specifying which metrics to include in the tornado plot. Supported options are "value" (value at tornado_time) and "cmax" (maximum output). Defaults to "value".
ref_data	Optional data frame providing reference (baseline) output values, formatted similarly to sens_data. If NULL, the function uses the mid-range parameter value as the baseline.
log_scale	Logical. If TRUE, the y-axis of the family-of-curves plot is displayed on a log scale.
color_low	Character. Color used for the lower parameter bound (default: "#1f77b4").
color_high	Character. Color used for the upper parameter bound (default: "#ff7f0e").
facet_scales	Character string passed to facet_grid(), defining how y-axis scales behave across facets (default = "free").

Value

A named list containing:

family_of_curves A ggplot2 object showing parameter sensitivity curves over time.

tornado A ggplot2 object showing the tornado plot of relative sensitivity.

Examples

```
library(rxode2)
library(tibble)
mod_ex <- RxODE({
  # Doses in mg
  # Time in hours
  ### Parameter values
  # Typical
```

```
POPCL = 5;
POPVC = 180;
POPQ = 7;
POPVP = 52;
POPKTR = 6;
FBIOPAR = 1;
# Covariate coefficients
POPIGFRCOV = 0.8;
POPPATCOVCLGR1 = 0.85;
POPPATCOVCLGR2 = 0.85;
POPPATCOVCLGR3 = 0.85;
POPPATCOVCLGR4 = 0.85;
# Random effects
PPVCL = 0;
PPVVC = 0;
PPVKTR = 0;
# Residual error
RUV = 0;
### Covariates
PATCOVCL = 1
if(POPN == 1){PATCOVCL = POPPATCOVCLGR1}
if(POPN == 2){PATCOVCL = POPPATCOVCLGR2}
if(POPN == 3){PATCOVCL = POPPATCOVCLGR3}
if(POPN == 4){PATCOVCL = POPPATCOVCLGR4}
IGFRCOV = (IGFR/112)^POPIGFRCOV
## Parameters
CL = POPCL * IGFRCOV * PATCOVCL * exp(PPVCL);
VC = POPVC * exp(PPVVC);
Q = POPQ;
VP = POPVP;
KTR = POPKTR * exp(PPVKTR);
### Explicit functions
Cc = Ac/VC;                 # nmol/L
Cp = Ap/VP;                 # nmol/L
### Initial conditions
At1(0) = 0;         # mg
At2(0) = 0;         # mg
At3(0) = 0;         # mg
At4(0) = 0;         # mg
At5(0) = 0;         # mg
At6(0) = 0;         # mg
Ad(0) = 0;          # mg
Ac(0) = 0;          # mg
Ap(0) = 0;          # mg
### ODEs
d/dt(At1) = - KTR*At1;
d/dt(At2) = KTR*At1 - KTR*At2;
d/dt(At3) = KTR*At2 - KTR*At3;
d/dt(At4) = KTR*At3 - KTR*At4;
d/dt(At5) = KTR*At4 - KTR*At5;
d/dt(At6) = KTR*At5 - KTR*At6;
d/dt(Ad) = KTR*At6 - KTR*Ad;
d/dt(Ac) = KTR*Ad - CL*Cc - Q*(Cc - Cp);
```

```
  d/dt(Ap) = Q*(Cc - Cp);
  FBIO = FBIOPAR
  f(At1) = FBIO*1000000/505;
  CHECKRUV = RUV;
  Cc_ResErr = Cc + RUV*Cc;
})
et_base <- tribble(
  ~id, ~time, ~evid, ~cmt, ~amt, ~addl, ~ii, ~IGFR, ~POPN,
  1,   0,     1,     1,    10,   2,     24,  112,   1
)
sens_loc <- sg_localsens_sim(
  model = mod_ex,
  params = c("POPCL", "POPVC"),
  stimes = seq(0, 168, 0.1),
  output = "Cc",
  perc = 0.9,
  cov = c("IGFR", "POPN"),
  et = et_base
)
plots <- sg_localsens_vis(
  sens_data = sens_loc,
  tornado_time = 168,
  metrics = c("value", "cmax"),
  log_scale = TRUE,
  color_low = "#4575b4",
  color_high = "#d73027",
  facet_scales = "free"
)

# Display plots
plots$family_of_curves
plots$tornado
```

`sg_modbuild` *Build estimation model scenarios*

Description

Constructs and exports multiple model configuration scenarios by combining population parameters, random effects, residual variability, occasion effects, and covariate assignments. Each scenario is defined by unique combinations of these model components and exported as `.mlxtran` files. Additionally, a summary CSV file describing all scenarios is generated.

Usage

```
sg_modbuild(
  mod_lst,
  data,
  headers,
```

```
  ruv_lst,
  theta_lst,
  re_lst,
  path,
  occ_lst,
  covs_lst = NULL,
  task_lst = NULL,
  opt_name = "Simurg",
  project_name = "my_project"
)
```

Arguments

mod_lst	List of model file paths or model identifiers used for scenario generation.
data	Character string. Path to the dataset used in model fitting or simulation.
headers	Predefined list of dataset column names (e.g., ID, TIME, DV, etc.) used by the modeling framework.
ruv_lst	List specifying residual unexplained variability (RUV) structures. Each element contains RUV properties (e.g., YNAME, ERR) and mapping to data.
theta_lst	List of tibbles describing population parameter properties. Each tibble should contain columns such as NAME, INIT, and EST.
re_lst	List of random effects (RE) specifications. Each element includes matrices init and est defining initial and estimated covariance structures.
path	Character. Directory path where output files (CSV summary and model files) will be written. Default is current working directory.
occ_lst	List of occasion effect matrices. Each element includes init and est matrices defining inter-occasion variability.
covs_lst	Optional list describing covariate-parameter relationships. Each element should include fields such as PAR, COVNAME, and EST indicating inclusion in the model.
task_lst	Optional list defining additional tasks or configuration options for each scenario. Default is NULL.
opt_name	Character. Optimization engine name (e.g., "Monolix", "Simurg"). Default "Simurg".
project_name	Character. Base name for the exported project files. Default "my_project".

Details

This function automates the construction of multiple model configurations for model evaluation, sensitivity analysis, or scenario testing. It expands parameter and variability definitions, constructs all possible combinations, and exports model specifications for external fitting tools.

Value

The function writes two types of outputs to the specified path:

- A CSV file (scenarios_info.csv) summarizing all generated scenarios with columns:

scenario_number Unique index of the scenario.

model Model file or path used in the scenario.

data Path to the dataset used.

theta Active population parameters and initial values used.

RUV Residual error structure(s) used.

RE Random effect and correlation terms included.

OCC Active occasion effects.

COVS Included covariate-parameter relationships.

- Individual `.mlxtran` model files for each generated scenario, named according to the pattern `<project_name>_<i>.mlxtran`.

Examples

```
library(dplyr)
folder_path <- system.file("extdata", package = "SimuRg")
mod_lst <- list(paste(folder_path, "/models/model_PK_1c.txt", sep = "/"),
                  paste(folder_path, "/models/model_PK_2c.txt", sep = ""))

### path to the dataset
data <- paste(folder_path, "datasets", "dspk-warf.csv", sep = "/")
re_lst_1 <- list(
              list(init = tribble(~Cl, ~Vd, ~ka, ~Vp, ~Q,
                                  1, 0, 0, 0, 0,
                                  0, 1, 0, 0, 0,
                                  0, 0, 1, 0, 0,
                                  0, 0, 0, 1, 0,
                                  0, 0, 0, 0, 1) %>% as.matrix(),
                   est = tribble(~Cl, ~Vd, ~ka, ~Vp, ~Q,
                                  TRUE, NA, NA, NA, NA,
                                  NA, TRUE, NA, NA, NA,
                                  NA, NA, TRUE, NA, NA,
                                  NA, NA, NA, TRUE, NA,
                                  NA, NA, NA, NA, TRUE) %>% as.matrix(),
                   block = tribble(~Cl, ~Vd, ~ka, ~Vp, ~Q,
                                    FALSE, NA, NA, NA, NA,
                                    NA, FALSE, NA, NA, NA,
                                    NA, NA, TRUE, NA, NA,
                                    NA, NA, NA, FALSE, NA,
                                    NA, NA, NA, NA, FALSE) %>% as.matrix())
           )
headers <- list(list(name = "ID", use = "identifier", type = NULL),
                list(name = "TIME", use = "time", type = NULL),
                list(name = "DV", use = "observation", type = "continuous"),
                list(name = "DVID", use = "observationtype", type = NULL),
                list(name = "ADM", use = "administration", type = NULL),
                list(name = "AMT", use = "amount", type = NULL),
                list(name = "EVID", use = "eventidentifier", type = NULL),
                list(name = "MDV", use = "missingdependentvariable", type = NULL),
                list(name = "AGE", use = "covariate", type = "continuous"),
                list(name = "AGE_centered", use = "covariate", type = "continuous"),
                list(name = "SEX", use = "covariate", type = "categorical"),
```

```
                list(name = "WEIGHT", use = "covariate", type = "continuous"),
                list(name = "BMI", use = "covariate", type = "continuous"))

ruv_lst_2 <- list(
  # structural model 1
 list(
   list(YNAME = "y1",
        DVID = 1,
        TRANS = "normal",
        PRED = "Cc",
        ERR = list("constant", "proportional", "combined1"),
         # options to test (can be length = 1, i.e., "constant" or "combined1")
        INIT = list(1, 1, c(1, 1)),
         # options to test (can be length = 1, i.e., 1 or c(1, 1))
        EST = list(TRUE, TRUE, c(TRUE, TRUE)))
        # options to test (can be length = 1, i.e., T or c(T, T))BLQM = NULL))
 ),
 # structural model 2
 list(
   list(YNAME = "y1",
        DVID = 1,
        TRANS = "normal",
        PRED = "Cc",
        ERR = list("constant", "proportional"),
        # options to test (can be length = 1, i.e., "constant" or "combined1")
        INIT = list(1, 1), # options to test (can be length = 1, i.e., 1 or c(1, 1))
        EST = list(TRUE, TRUE))
         # options to test (can be length = 1, i.e., T or c(T, T))BLQM = NULL))
 )
)

theta_lst_2 <- list(
 # structural model 1
 tribble(~NAME, ~TRANS, ~INIT, ~LB, ~UB, ~EST,
         "Cl", "logNormal", 0.2, NA, NA, TRUE,
         "Vd", "logNormal", 20, NA, NA, TRUE,
         "ka", "logNormal", 0.2, NA, NA, TRUE),
 # structural model 2
 tribble(~NAME, ~TRANS, ~INIT, ~LB, ~UB, ~EST,
         "Cl", "logNormal", 0.2, NA, NA, TRUE,
         "Vd", c("Normal", "logNormal"), c(10, 20, 30), NA, NA, TRUE,
         "ka", "logNormal", c(0.2, 0.1, 0.5), NA, NA, TRUE,
         # INIT could be length > 1
         "Vp", c("Normal", "logNormal"), 10, NA, NA, TRUE,
         "Q", "logNormal", 5, NA, NA, TRUE))

path <- tempdir()
sg_modbuild(
 mod_lst = mod_lst[1],
 data = data,
 headers = headers,
 ruv_lst = ruv_lst_2,
 theta_lst = theta_lst_2,
```

```
 re_lst = re_lst_1,
 occ_lst = re_lst_1,
 covs_lst = NULL,
 path = path,
 project_name = "tests_test_project"
)
clr_files <- list.files(path, full.names = TRUE)
unlink(clr_files, recursive = TRUE, force = TRUE)
```

`sg_modcomp` *Returns dataframe with summary statistics for model comparison*

Description

Returns dataframe with summary statistics for model comparison

Usage

```
sg_modcomp(fpath_i, run_id = 1)
```

Arguments

`fpath_i`	String, GFO, or a named list with a `GFO` component (and optionally `GCO`). If a string is given, the path to a `.RData` or `.json` file with a fit object is expected.
`run_id`	Integer. Tested model ID. Default is 1.

Value

Data frame with model comparison metrics derived from GFO `$OFV` and related fit outputs.

Examples

```
fpath_i <- system.file("extdata", "simurg_object", "Warfarin_PK.RData", package = "SimuRg")
sum_tab <- sg_modcomp(fpath_i, 3)
print(sum_tab)
```

`sg_multistart` *Build multistart scenarios with varying initial values*

Description

Generates multiple scenarios with identical model structure (model, RUV, RE, OCC, COVS) and different sampled initial values for population parameters.

For each start, initial values of estimated parameters are sampled uniformly within specified intervals, and a corresponding `.mlxtran` file is generated. A summary table describing all multistart scenarios is written to disk.

Usage

```
sg_multistart(
  mod,
  data,
  headers,
  ruv,
  theta,
  re,
  occ,
  path,
  covs_lst = NULL,
  opt_name = "Simurg",
  project_name = "multistart_project",
  n_starts = 10,
  theta_intervals = NULL,
  interval_factor = c(0.2, 5),
  vary_params = NULL,
  seed = NULL
)
```

Arguments

`mod`	Model file path or model identifier used for scenario generation.
`data`	Character string. Path to the dataset used in model fitting or simulation.
`headers`	Predefined list of dataset column names (e.g., ID, TIME, DV, etc.) used by the modeling framework.
`ruv`	Residual unexplained variability (RUV) structure. Each element contains RUV properties (e.g., `YNAME`, `ERR`) and mapping to data.
`theta`	Tibble describing population parameter properties. Tibble should contain columns such as `NAME`, `INIT`, and `EST`.
`re`	Random effects (RE) specification. Includes matrices `init` and `est` defining initial and estimated covariance structures.

occ	Occasion effect matrices. Includes init and est matrices defining inter-occasion variability.
path	Character. Directory path where output files (CSV summary and model files) will be written. Default is current working directory.
covs_lst	Optional list describing covariate-parameter relationships. Each element should include fields such as PAR, COVNAME, and EST indicating inclusion in the model.
opt_name	Character. Optimization engine name (e.g., "Monolix", "Simurg"). Default "Simurg".
project_name	Character. Base name for the exported project files. Default "my_project".
n_starts	Number of multistart runs (number of initial value scenarios).
theta_intervals	Optional data frame specifying sampling intervals for initial values. Must contain columns NAME, lower, and upper. If NULL, intervals are derived from interval_factor.
interval_factor	Numeric vector of length 2 specifying multiplicative lower and upper bounds factors relative to original INIT values (default c(0.2, 5)).
vary_params	List of parameters which inital values are varied. Default value implies all parameters are varied. If theta_interval is specified, list of parameters is taken from theta_interval
seed	Optional random seed for reproducible sampling of initial values. If NULL, the global RNG state is not modified.

Details

For each start:

- Population parameter initial values INIT for estimated parameters are sampled uniformly.
- A scenario summary is stored.
- The corresponding .mlxtran project files are written.

If seed is provided, the function temporarily sets the random number generator state and restores it upon exit.

Value

The function writes two types of outputs to the specified path:

- A CSV file (scenarios_info.csv) summarizing all generated scenarios with columns:

  scenario_number Unique index of the scenario.

  model Model file or path used in the scenario.

  data Path to the dataset used.

  theta Active population parameters and initial values used.

  RUV Residual error structure(s) used.

  RE Random effect and correlation terms included.

  OCC Active occasion effects.

COVS Included covariate-parameter relationships.

- Individual `.mlxtran` model files for each generated scenario, named according to the pattern `<project_name>_<i>.mlxtran`.

Examples

```
library(dplyr)
folder_path <- system.file("extdata", package = "SimuRg")

mod <- paste(folder_path, "/models/model_PK_1c.txt", sep = "/")

data <- paste(folder_path, "datasets", "dspk-warf.csv", sep = "/")
re <- list(init = tribble(~Cl, ~Vd, ~ka, ~Vp, ~Q,
                                  1, 0, 0, 0, 0,
                                  0, 1, 0, 0, 0,
                                  0, 0, 1, 0, 0,
                                  0, 0, 0, 1, 0,
                                  0, 0, 0, 0, 1) %>% as.matrix(),
                   est = tribble(~Cl, ~Vd, ~ka, ~Vp, ~Q,
                                 TRUE, NA, NA, NA, NA,
                                 NA, TRUE, NA, NA, NA,
                                 NA, NA, TRUE, NA, NA,
                                 NA, NA, NA, TRUE, NA,
                                 NA, NA, NA, NA, TRUE) %>% as.matrix(),
                   block = tribble(~Cl, ~Vd, ~ka, ~Vp, ~Q,
                                   FALSE, NA, NA, NA, NA,
                                   NA, FALSE, NA, NA, NA,
                                   NA, NA, TRUE, NA, NA,
                                   NA, NA, NA, FALSE, NA,
                                   NA, NA, NA, NA, FALSE) %>% as.matrix())

headers <- list(list(name = "ID", use = "identifier", type = NULL),
                list(name = "TIME", use = "time", type = NULL),
                list(name = "DV", use = "observation", type = "continuous"),
                list(name = "DVID", use = "observationtype", type = NULL),
                list(name = "ADM", use = "administration", type = NULL),
                list(name = "AMT", use = "amount", type = NULL),
                list(name = "EVID", use = "eventidentifier", type = NULL),
                list(name = "MDV", use = "missingdependentvariable", type = NULL),
                list(name = "AGE", use = "covariate", type = "continuous"),
                list(name = "AGE_centered", use = "covariate", type = "continuous"),
                list(name = "SEX", use = "covariate", type = "categorical"),
                list(name = "WEIGHT", use = "covariate", type = "continuous"),
                list(name = "BMI", use = "covariate", type = "continuous"))

theta <- tribble(~NAME, ~TRANS, ~INIT, ~LB, ~UB, ~EST,
                   "Cl", "logNormal", 0.2, NA, NA, TRUE,
                   "Vd", "logNormal", 20, NA, NA, TRUE,
                   "ka", "logNormal", 0.2, NA, NA, TRUE)

theta_intervals <- list(
  Cl = c(0.25*theta$INIT[theta$NAME == "Cl"], 4*theta$INIT[theta$NAME == "Cl"]),
```

```
  #Vd = c(0.5*theta$INIT[theta$NAME == "Vd"], 2*theta$INIT[theta$NAME == "Vd"]),
  ka   = c(1.25*theta$INIT[theta$NAME == "ka"], 1.5*theta$INIT[theta$NAME == "ka"])
)

ruv <- list(YNAME = "y1",
            DVID = 1,
            TRANS = "normal",
            PRED = "Cc",
            ERR = "proportional",
            INIT = 1,
            EST = TRUE)

n_starts <- 20

path <- tempdir()
sg_multistart(
  mod = mod,
  data = data,
  headers = headers,
  ruv = ruv,
  theta = theta,
  re = re,
  occ = re,
  n_starts = n_starts,
  theta_intervals = theta_intervals,
  path = path,
  project_name = "multistart_test_project"
)
clr_files <- list.files(path, full.names = TRUE)
unlink(clr_files, recursive = TRUE, force = TRUE)
```

`sg_parsum` *Extract parameter summary (theta, eta, SE, RSE, CI, CV, shrinkage)*

Description

Extract parameter summary (theta, eta, SE, RSE, CI, CV, shrinkage)

Usage

```
sg_parsum(fpath_i, addOFV = TRUE)
```

Arguments

`fpath_i`	String, GFO, or a named list with a `GFO` component (and optionally `GCO`). If a string is given, the path to a `.RData` or `.json` file with a fit object is expected.
`addOFV`	Logical. If `TRUE`, information about OFV and AIC of the model will be added. Default is `TRUE`.

Value

Table derived from GFO $SUMTAB (and $OFV when addOFV = TRUE).

Examples

```
fpath_i <- system.file("extdata", "simurg_object", "Warfarin_PK.RData", package = "SimuRg")
sum_tab <- sg_parsum(fpath_i)
print(sum_tab)
```

---

sg_predist_sim *Perform simulations for prediction distribution plots*

Description

Perform simulations for prediction distribution plots

Usage

```
sg_predist_sim(
  fpath_i,
  gco = NULL,
  model = NULL,
  time_col = "TIME",
  outputs = NULL,
  npop = 500
)
```

Arguments

fpath_i	String, GFO, or a named list with a GFO component (and optionally GCO). If a string is given, the path to a .RData or .json file with a fit object is expected.
gco	Generalized control object (see GMO). Either an R list or path to Rdata/json file. Either gco or model should be specified for the simulation model specification. Default is NULL
model	GMO. RxODE2 model to simulate from.
time_col	String. The column to use as a time column. Currently, can be only TIME. Default is TIME.
outputs	Vector of strings. Names of the model variables to output. If NULL, all variables returned. Default is NULL.
npop	Integer. Number of population replicates. Default is 1.

Value

GSO: a data frame with simulation results.

See Also

GFO, GMO, GSO, `sg_sim()`, `sg_vpc_sim()`

Examples

```
library(rxode2)
fpath_i <- system.file("extdata", "simurg_object", "Warfarin_PK.RData", package = "SimuRg")
mod_fin <- rxode2({
 # Differential equations
 ka = ka_pop*exp(omega_ka)
 V = V_pop
 Cl = Cl_pop*exp(omega_Cl)
 d/dt(Ad) = -ka * Ad
 d/dt(Ac) = ka * Ad - Cl/V * Ac
 # Concentration calculations
 Cc = Ac / V
})
predist_sim <- sg_predist_sim(fpath_i, model = mod_fin, outputs = "Cc")
predist_sim
```

---

sg_predist_vis *Generate prediction distribution visualizations from sg_predist_sim() simulation results*

Description

Generate prediction distribution visualizations from sg_predist_sim() simulation results

Usage

```
sg_predist_vis(
  fpath_i,
  ds_sim,
  time_col = "TIME",
  lab_x = "TIME",
  lab_y = NULL,
  dt_obs_fl = FALSE,
  logy = FALSE,
  legend_fl = TRUE,
  pred_interval = "80%"
)
```

Arguments

`fpath_i`	String, GFO, or a named list with a `GFO` component (and optionally `GCO`). If a string is given, the path to a `.RData` or `.json` file with a fit object is expected.

ds_sim	GSO. Data frame with simulation results in long format (columns ID, TIME, VAR, VALUE), typically from sg_predist_sim().
time_col	String. The column to use as a time column. Currently, can be only TIME. Default is TIME.
lab_x	Character. X-axis label. Default is "TIME" with its own output variable (the VAR value, with any _ResErr suffix removed).
lab_y	Character. Y-axis label. If NULL (default), each plot is labelled
dt_obs_fl	Logical. If TRUE, observed data points are overlaid on the simulation plots. Default is FALSE.
logy	Logical. If TRUE, y-axis is displayed on a logarithmic scale. Default is FALSE.
legend_fl	Logical. If FALSE, the legend is hidden. Default is TRUE.
pred_interval	Character. Prediction interval to display. Options are "95%", "90%", "80%", "50%". Default is "80%".

Value

A list of ggplot objects, one per output variable in the simulation dataset.

See Also

sg_predist_sim(), sg_vpc_vis()

Examples

```
library(rxode2)
fpath_i <- system.file("extdata", "simurg_object", "Warfarin_PK.RData", package = "SimuRg")
mod_fin <- rxode2({
 # Differential equations
 ka = ka_pop*exp(omega_ka)
 V = V_pop
 Cl = Cl_pop*exp(omega_Cl)
 d/dt(Ad) = -ka * Ad
 d/dt(Ac) = ka * Ad - Cl/V * Ac
 # Concentration calculations
 Cc = Ac / V
})
predist_sim <- sg_predist_sim(fpath_i, model =mod_fin, outputs = "Cc")
predist_vis <- sg_predist_vis(fpath_i,predist_sim)
predist_vis
```

sg_sim *Perform simulations from an rxode2 model*

Description

Runs simulations using a GMO with explicit R objects for parameters (theta), variance–covariance (thetamat), omega, and sigma. Argument layout is described in GSI.

Usage

```
sg_sim(
  model,
  et,
  stimes = NULL,
  outputs = NULL,
  theta = NULL,
  omega = NULL,
  sigma = NULL,
  thetamat = NULL,
  covs = NULL,
  npop = 1,
  nsub = 1,
  aggr = NULL,
  addcov = TRUE,
  keep = NULL,
  scale = NULL,
  covint = "locf",
  inits = NULL,
  byID = NULL,
  byPOP = NULL,
  shared = NULL,
  ncores = 1,
  atol = 1e-08,
  rtol = 1e-06,
  fpath_i = NULL,
  ...
)
```

Arguments

model	GMO. RxODE2 model to simulate from.
et	Data.frame. Event table; a component of GSI.
stimes	Numeric vector. Sampling time points; a component of GSI. Default is NULL.
outputs	Vector of strings. Names of the model variables to output. If NULL, all variables returned. Default is NULL.

`theta` Named vector or data frame. Population or baseline parameter values; a component of GSI. For sensitivity analysis, parameters listed in `params` are replaced by sampled values. Default is `NULL`.

`omega` Named matrix or vector. Inter-individual variability matrix; a component of GSI.

`sigma` Named matrix. Residual error variance matrix or its Cholesky decomposition; a component of GSI. Default is `NULL`.

`thetamat` Named variance–covariance matrix for parameters on the normal scale; a component of GSI. Default is `NULL`.

`covs` A list of covariate structures. Each element should be a list containing covariate relationship definitions with the following structure:

- `PAR` - string, name of the parameter to which covariate effect is applied
- `COVNAME` - string, name of the covariate column in the dataset
- `FUNC` - string, functional form of the covariate effect
- `TRANS` - string, transformation applied to covariate
- `REF` - numeric, reference value for categorical covariates
- `INIT` - numeric, initial value for the covariate effect parameter
- `EST` - logical, whether to estimate the covariate effect Can be `NULL`, if no covariates are used. Default is `NULL`

`npop` Integer. Number of population replicates. Default is 1.

`nsub` Integer. Number of subjects sampled per population (omega/sigma matrices per ID). Default is 1.

`aggr` A string that specifies the aggregation type. Set to `NULL` for no aggregation, `ID` for aggregation by time, population and ID, `NPOP` for aggregation by population and time, `TIME` for aggregation by time. Default is `NULL`.

`addcov` Logical. If `TRUE`, columns with covariate values will be added to the resulting dataset. Default is `TRUE`.

`keep` Vector of strings. Columns of event table to keep in the output data frame. Default is `NULL`.

`scale` A numeric named vector. Scaling for ODE parameters of the system. The names must correspond to the parameter identifiers in the ODE specification. Each of the ODE variables will be divided by the scaling factor. Default is `NULL`.

`covint` String. Specifies the interpolation method for time-varying covariates. When solving ODEs it often samples times outside the sampling time specified in event table. When this happens, the time varying covariates are interpolated. Currently this can be:

- `"linear"` interpolation, which interpolates the covariate by solving the line between the observed covariates and extrapolating the new covariate value
- `"locf"` last observation carried forward
- `"NOCB"` next observation carried backward. This is the same method that NONMEM uses
- `"midpoint"` last observation carried forward to midpoint; next observation carried backward to midpoint

	Default is "locf"
inits	A named vector specifying initial conditions for model variables; Default is NULL.
byID	logical. If TRUE, replicate populations/subjects per event-table ID. Default NULL is treated as TRUE.
byPOP	logical. When both thetamat and omega are used: if TRUE, replicate subjects by population. Default NULL is treated as TRUE.
shared	logical. If TRUE, one shared population draw for all IDs. Default NULL is treated as TRUE.
ncores	Integer. Number of cores used for calculations. Default is 1.
atol	A numeric absolute tolerance used by the ODE solver to determine if a good solution has been achieved. This is also used in the solved linear model to check if prior doses do no add anything to the solution. Default is 1e-8.
rtol	Numeric. A numeric relative tolerance used by the ODE solver to determine if a good solution has been achieved. This is also used in the solved linear model to check if prior doses do not add anything to the solution. Default is 1e-6.
fpath_i	String, GFO, or a named list with a GFO component (and optionally GCO). If a string is given, the path to a .RData or .json file with a fit object is expected.
...	optional arguments passed to rxode2::rxSolve (e.g. method, maxsteps).

Details

Index columns ID, POPN, and sim.id are included depending on what is used:

No uncertainty, no variability Only ID (and TIME, VAR, VALUE)

Only uncertainty (thetamat) POPN and ID

Only variability (omega) sim.id and ID

Uncertainty and variability POPN, sim.id, and ID

Other columns:

TIME Timepoint of the simulations

VAR Names of the output variables

VALUE Optional. Simulated value. Returned when no aggregation applied

mean, median, ... Optional. Aggregated statistic of simulated values. Returned when aggregation is applied

COV1, ...COVn Optional. Covariates value. Returned when addcov is TRUE

KEEP1, ...KEEPn Optional. Columns that were specified in the keep argument

Event table can use either id or ID. Parameters can be fixed per ID by including parameter names (e.g. *_pop, omega_*) as columns in the event table.

Value

GSO: a data frame with simulation results (long format).

Examples

```
library(rxode2)
library(dplyr)
library(tibble)

# Bundled [GMO] and [GSI] (see ?gmo_pk1c, ?gsi_pk1c)
sim_gmo <- do.call(sg_sim, c(list(model = gmo_pk1c), gsi_pk1c))
head(sim_gmo)

# Base 1-compartment PK model (no covariate)
mod <- rxode2::rxode2({
  ka_pop = 0.1;
  Vd_pop = 10;
  CL_pop = 0.5;

  omega_ka = 0;
  omega_Vd = 0;
  omega_CL = 0;

  Cc_b = 0;
  ka_tv = exp(ka_pop);
  Vd_tv = exp(Vd_pop);
  CL_tv = exp(CL_pop);

  ka = ka_tv * exp(omega_ka);
  Vd = Vd_tv * exp(omega_Vd);
  CL = CL_tv * exp(omega_CL);

  Cc = Ac / Vd;

  Ad(0) = 0;
  Ac(0) = 0;

  d/dt(Ad) = -ka * Ad;
  d/dt(Ac) = ka * Ad - CL * Cc;

  Cc_ResErr = Cc * (1 + Cc_b);
})

# Model with covariate: WTBL on Vd (Vd_tv = exp(Vd_pop) * (WTBL/70)^beta_WTBL_Vd_pop)

mod_cov <- rxode2::rxode2({
  ka_pop = 0.1;
  Vd_pop = 10;
  CL_pop = 0.5;

  beta_WTBL_Vd_pop = 0;

  omega_ka = 0;
  omega_Vd = 0;
  omega_CL = 0;
```

```
  Cc_b = 0;

  ka_tv = exp(ka_pop);
  Vd_tv = exp(Vd_pop) * (WTBL/70)^beta_WTBL_Vd_pop;
  CL_tv = exp(CL_pop);

  ka = ka_tv * exp(omega_ka);
  Vd = Vd_tv * exp(omega_Vd);
  CL = CL_tv * exp(omega_CL);

  Cc = Ac / Vd;

  Ad(0) = 0;
  Ac(0) = 0;

  d/dt(Ad) = -ka * Ad;
  d/dt(Ac) = ka * Ad - CL * Cc;

  Cc_ResErr = Cc * (1 + Cc_b);
})

et_test <- tibble::tribble(
  ~ID, ~TIME, ~EVID, ~CMT, ~AMT,
  1, 0, 1, 1, 10,
  2, 0, 1, 1, 50
)

stimes_test <- seq(0, 24, 0.1)
output_test <- c("Ac", "Ad", "Cc", "Cc_ResErr")
theta_test <- c(ka_pop = log(0.1), Vd_pop = log(10), CL_pop = log(0.5))

thetamat_test <- matrix(c(0.05, 0.01, 0, 0.01, 0.05, 0, 0, 0, 0.05), nrow = 3, byrow = TRUE)
rownames(thetamat_test) <- colnames(thetamat_test) <- c("ka_pop", "Vd_pop", "CL_pop")

omega_test <- matrix(c(0.2, 0.05, 0, 0.05, 0.2, 0, 0, 0, 0.2), nrow = 3, byrow = TRUE)
rownames(omega_test) <- colnames(omega_test) <- c("omega_ka", "omega_Vd", "omega_CL")

sigma_test <- matrix(0.1); rownames(sigma_test) <- colnames(sigma_test) <- "Cc_b"

# No variability
sim1 <- sg_sim(model = mod, et = et_test, stimes = stimes_test, theta = theta_test,
               outputs = output_test)

# Population uncertainty, single scenario (ID)
sim2a <- sg_sim(model = mod, et = dplyr::filter(et_test, ID == 1),
                stimes = stimes_test, theta = theta_test, thetamat = thetamat_test,
                npop = 10)

# Population uncertainty, multiple IDs, byID = TRUE (populations are replicated
# for each scenario (ID)),
# shared = TRUE (the same populations are used for each scenario (ID))
sim2b <- sg_sim(model = mod, et = et_test, stimes = stimes_test, theta = theta_test,
               thetamat = thetamat_test, npop = 10, byID = TRUE, shared = TRUE)
```

```
# Population uncertainty, multiple IDs, byID = FALSE (one solve over full event table;
# npop must equal n(ID); ID - virtual subject)
sim2c <- sg_sim(model = mod, et = et_test, stimes = stimes_test, theta = theta_test,
               thetamat = thetamat_test, npop = nrow(et_test), byID = FALSE)


# Between-subject variability (BSV), multiple IDs, byID = TRUE (subjects are
# replicated for each scenario (ID)), shared = FALSE (separate subjects per ID)
sim3a <- sg_sim(model = mod, et = et_test, stimes = stimes_test, theta = theta_test,
                omega = omega_test, nsub = 10, byID = TRUE, shared = FALSE)


# Between-subject variability (BSV), byID = FALSE (one solve over full event table;
# nsub must equal n(ID); ID - virtual subject)
sim3b <- sg_sim(model = mod, et = et_test, stimes = stimes_test, theta = theta_test,
                omega = omega_test, nsub = nrow(et_test), byID = FALSE)


# BSV + uncertainty: byID = TRUE (populations/subjects are replicated for each
# scenario (ID)), byPOP = TRUE (subjects are replicated for each population),
# shared = FALSE (separate populations/subjects per ID)
sim4a <- sg_sim(model = mod, et = et_test, stimes = stimes_test, theta = theta_test,
                thetamat = thetamat_test, npop = 10, omega = omega_test, nsub = 5,
                byID = TRUE, byPOP = TRUE, shared = FALSE)

# BSV + uncertainty: byID = TRUE (populations/subjects are replicated for each
# scenario (ID)), byPOP = FALSE (subjects are not replicated between population;
# npop = nsub)
sim4b <- sg_sim(model = mod, et = et_test, stimes = stimes_test, theta = theta_test,
                thetamat = thetamat_test, npop = nrow(et_test), omega = omega_test,
                nsub = nrow(et_test), byID = TRUE, byPOP = FALSE)

# BSV + uncertainty: byID = FALSE (one solve over full event table; nsub and/or
# npop must equal n(ID); ID - virtual subject), byPOP = TRUE (subjects are
# replicated for each population), shared = FALSE (separate populations/subjects
# per ID)
sim4c <- sg_sim(model = mod, et = et_test, stimes = stimes_test, theta = theta_test,
                thetamat = thetamat_test, npop = 10, omega = omega_test, nsub = 5,
                byID = FALSE, byPOP = TRUE, shared = FALSE)

# BSV + uncertainty: byID = FALSE (one solve over full event table; nsub and/or
# npop must equal n(ID); ID - virtual subject), byPOP = FALSE (subjects are not
# replicated between population; npop = nsub)
sim4d <- sg_sim(model = mod, et = et_test, stimes = stimes_test, theta = theta_test,
                thetamat = thetamat_test, npop = nrow(et_test), omega = omega_test,
                nsub = nrow(et_test), byID = FALSE, byPOP = FALSE)

# Parameter override in event table
sim4 <- sg_sim(model = mod, et = dplyr::filter(et_test, ID == 1) %>%
                                 dplyr::mutate(Vd_pop = 2),
               stimes = stimes_test, theta = theta_test, thetamat = thetamat_test,
               npop = 1)
```

```
# Covariate (WTBL): pass covs and keep so WTBL is used and returned
theta_cov <- c(theta_test, beta_WTBL_Vd_pop = 0.5)
sim5 <- sg_sim(model = mod_cov, et = dplyr::filter(et_test, ID == 1) %>%
                                     dplyr::mutate(WTBL = 70),
               stimes = stimes_test, theta = theta_cov, covs = c("WTBL"),
               keep = c("WTBL"),
               thetamat = thetamat_test, npop = 5)
```

---

sg_sim_tp *Time-profile plotting with optional summary bands*

Description

Creates a ggplot time-profile plot from longitudinal data with optional summarization (mean/median/geometric mean), variance bands (SD/SE/IQR), percentile ribbons, faceting, and log scaling.

Usage

```
sg_sim_tp(
  ds_i,
  time_col = "TIME",
  val_col = "VALUE",
  group_i = "VAR",
  bands_i = NULL,
  cent_i = NULL,
  vrns_i = NULL,
  lperc_i = NULL,
  uperc_i = NULL,
  add_points = 0,
  col_i = NULL,
  fill_i = NULL,
  lty_i = NULL,
  shp_i = NULL,
  grid_i = NULL,
  wrap_i = NULL,
  free_stat = "free",
  wrap_ncol = NULL,
  wrap_nrow = NULL,
  min_x = NA,
  max_x = NA,
  min_y = NA,
  max_y = NA,
  log_y = FALSE,
  log_x = FALSE
)
```

Arguments

ds_i	Data.frame. The data frame with source data.
time_col	String. The column to use as a time column. Currently, can be only TIME. Default is TIME.
val_col	String. Name of value column. Default is VALUE.
group_i	String. Primary grouping variable for lines. Default is 'VAR'.
bands_i	vector of characters. Character vector of length 2 with column names for custom ymin/ymax ribbons. Default is NULL
cent_i	string. Central tendency measure. One of 'mean', 'median', 'geom_mean', or NULL for raw values. Default is NULL
vrns_i	string. Variance band type. One of 'SD', 'SE', 'IQR', or NULL. Default is NULL
lperc_i	numeric. Lower percentile for percentile ribbons (must be provided together with uperc_i). Default is NULL
uperc_i	numeric. Upper percentile for percentile ribbons (must be provided together with lperc_i). Default is NULL
add_points	numeric. Size of points to add to the plot. If > 0, points will be added. Default is 0
col_i	String. Column name for color
fill_i	String. Column name for fill aesthetic. Default is NULL
lty_i	String. Column name for linetype aesthetic. Default is NULL.
shp_i	String. Column name for shape aesthetic. Default is NULL.
grid_i	string. Faceting formula for facet_grid (e.g., '~VAR' or 'A~B'). Default is NULL
wrap_i	String. Faceting formula for facet_wrap. Default is NULL.
free_stat	String. Facet scaling option. One of "free", "free_x", "free_y", or "fixed". Default is 'free'.
wrap_ncol	Integer. Number of columns for facet_wrap. Default is NULL.
wrap_nrow	Integer. Number of rows for facet_wrap. Default is NULL.
min_x	Numeric. X-axis minimum limit. Default is NA.
max_x	Numeric. X-axis maximum limit. Default is NA.
min_y	Numeric. Y-axis minimum limit. Default is NA.
max_y	Numeric. Y-axis maximum limit. Default is NA.
log_y	Logical. If TRUE, a logarithmic scale is applied to y-axis. Default is FALSE.
log_x	Logical. If TRUE, a logarithmic scale is applied to x-axis. Default is FALSE.

Value

A ggplot object with lines and optional ribbons/points/facets.

Examples

```
make_extended_mock_data <- function() {
 data.frame(
   TIME = rep(1:4, times = 6),
   VALUE = rnorm(24, mean = 10, sd = 2),
   VAR = rep(rep(c("A", "B"), each = 4), times = 3),
   Regimen = rep(c("R1", "R2", "R3"), each = 8),
   CAT1 = rep(c("Cat1", "Cat2"), each = 12)
 )
}
ds_sim <- make_extended_mock_data()
p <- sg_sim_tp(ds_i = ds_sim, group_i = 'VAR', col_i = 'VAR', fill_i = 'VAR',
               wrap_i = '~VAR', wrap_ncol = 2)
p
```

`sg_translator` *Translate structural models between rxode2 and MLXTRAN syntax*

Description

Converts structural pharmacometric model files between rxode2 (R package) and MLXTRAN (Monolix) syntax. Supports ODE-based models and Monolix pkmodel macros.

Usage

```
sg_translator(
  input_path,
  to,
  output_path,
  dm_list = NULL,
  regressors = NULL,
  output_vars = NULL,
  macros = TRUE,
  stiff = TRUE
)
```

Arguments

`input_path`	Character string. Path to the input structural model file (.txt).
`to`	Character string. Target format: `"mlxtran"` or `"rxode"`.
`output_path`	Character string. Path to write the translated model file (.txt).
`dm_list`	List of dosing macros with `cmt` and `adm` vectors. For example, `list(cmt = 2, adm = 1)` or `list(cmt = c(2, 1), adm = c(1, 2))`. Used to generate `iv()` dosing macros in MLXTRAN output.
`regressors`	Character vector of regressor variable names. For example, `c("WTBL", "ADAN")` generates `WTBL = {use = regressor}` in MLXTRAN.

output_vars	Character vector of output variable names for the MLXTRAN OUTPUT: section. For example, c("Cc_mgL", "CSF1_ngmL") generates output = {Cc_mgL, CSF1_ngmL}. Required when to = "mlxtran"; ignored when to = "rxode".
macros	Logical. If TRUE, attempt to convert rxode2 model to pkmodel macro format in MLXTRAN (only for simple 1- or 2-compartment oral PK models). Default TRUE.
stiff	Logical. If TRUE, add odeType = stiff to the MLXTRAN EQUATION section. Default TRUE.

Details

The function detects the source format automatically from file content.

rxode2 format uses # [INPUT], # [MODEL], # [OUTPUT] section markers, d/dt(X) ODE notation, f(X) for bioavailability, and alag(X) for lag time.

MLXTRAN format uses [LONGITUDINAL], PK:/EQUATION:/OUTPUT: sections, ddt_X ODE notation, compartment()/iv() dosing macros, and optionally pkmodel() macros.

When converting rxode2 to MLXTRAN with macros = TRUE, the function checks if the model structure matches a simple pkmodel() pattern (1- or 2-compartment with first-order oral absorption). If not, it falls back to full ODE format with compartment()/iv() macros.

MLXTRAN models using absorption()/elimination()/peripheral() macros (type 2) are expanded to explicit ODEs when converted to rxode2.

Value

Invisibly returns the translated model as a character string. The translated model is also written to output_path.

Examples

```
# Convert rxode2 model to MLXTRAN ODE format
rxode_model_hv_iv <- system.file("extdata", "models", "rxode",
                                 "model_PK_hv_iv.txt", package = "SimuRg")
rxode_model_1c <- system.file("extdata", "models", "rxode",
                              "model_PK_1c.txt", package = "SimuRg")
monolix_model_hv_iv <- system.file("extdata", "models", "monolix",
                                   "model_PK_hv_iv.txt", package = "SimuRg")
monolix_model_1c <- system.file("extdata", "models", "monolix",
                                "model_PK_1c.txt", package = "SimuRg")

path_to_save <- tempdir()

sg_translator(rxode_model_hv_iv, to = "mlxtran",
              output_path = normalizePath(file.path(path_to_save,
                                                    "model_PK_hv_iv_mlx.txt"),
                                          mustWork = FALSE),
              output_vars = "Cc_mgL",
              dm_list = list(cmt = 1, adm = 1), stiff = TRUE)

# Convert rxode2 model to MLXTRAN with pkmodel macros
```

```
sg_translator(rxode_model_1c, to = "mlxtran",
              output_path = normalizePath(file.path(path_to_save,
                                                    "model_PK_1c_mlx.txt"),
                                          mustWork = FALSE),
              output_vars = "Cc_nM", macros = TRUE)

# Convert MLXTRAN model to rxode2
sg_translator(monolix_model_hv_iv, to = "rxode",
              output_path = normalizePath(file.path(path_to_save,
                                                    "model_PK_hv_iv_rx.txt"),
                                          mustWork = FALSE))
```

`sg_vpc_sim` *Perform simulations for VPC plot*

Description

Perform simulations for VPC plot

Usage

```
sg_vpc_sim(
  fpath_i,
  gco = NULL,
  model = NULL,
  time_col = "TIME",
  outputs = NULL,
  npop = 100
)
```

Arguments

`fpath_i`	String, GFO, or a named list with a `GFO` component (and optionally `GCO`). If a string is given, the path to a `.RData` or `.json` file with a fit object is expected.
`gco`	GCO. Either an R list or path to an RData/json file. Either `gco` or `model` should be specified for model specification. Default is `NULL`.
`model`	GMO. RxODE2 model used when `gco` is not supplied. Default is `NULL`.
`time_col`	String. The column to use as a time column. Currently, can be only `TIME`. Default is `TIME`.
`outputs`	Vector of strings. Names of the model variables to output. If `NULL`, all variables returned. Default is `NULL`.
`npop`	Integer specifying the number of virtual subjects to simulate per original individual. Higher values provide more robust percentile estimates but increase computation time. Default is `100`

Value

GSO: a data frame with simulation results.

See Also

GCO, GMO, GSI, GSO, sg_sim()

Examples

```
library(rxode2)
library(tibble)
fpath_i <- system.file("extdata", "simurg_object", "Warfarin_PK.RData", package = "SimuRg")
# Simulate VPC from with generalized model object
mod <- rxode2::rxode2({
  ka_pop = 0.1;
  Vd_pop = 10;
  Cl_pop = 0.5;

  omega_ka = 0;
  omega_Vd = 0;
  omega_Cl = 0;

  Cc_b = 0;
  ka_tv = exp(ka_pop);
  Vd_tv = exp(Vd_pop);
  Cl_tv = exp(Cl_pop);

  ka = ka_tv * exp(omega_ka);
  Vd = Vd_tv * exp(omega_Vd);
  Cl = Cl_tv * exp(omega_Cl);

  Cc = Ac / Vd;

  Ad(0) = 0;
  Ac(0) = 0;

  d/dt(Ad) = -ka * Ad;
  d/dt(Ac) = ka * Ad - Cl * Cc;

  Cc_ResErr = Cc * (1 + Cc_b);
})

sg_vpc_sim(fpath_i, model=mod, outputs = "Cc_ResErr")
# Make VPC plot with the generalized control object
model <- system.file("extdata", "models", "rxode", "model_PK_1c.txt", package = "SimuRg")
data  <- system.file("extdata", "datasets", "dspk-warf.csv", package = "SimuRg")
headers <- list(list(name = "ID", use = "identifier", type = NULL),
             list(name = "TIME", use = "time", type = NULL),
                          list(name = "DV", use = "observation", type = "continuous"),
                          list(name = "DVID", use = "observationtype", type = NULL),
                          list(name = "ADM", use = "administration", type = NULL),
                          list(name = "AMT", use = "amount", type = NULL),
```

```
                          list(name = "EVID", use = "eventidentifier", type = NULL),
                        list(name = "MDV", use = "missingdependentvariable", type = NULL),
                          list(name = "AGE", use = "covariate", type = "continuous"),
                       list(name = "AGE_centered", use = "covariate", type = "continuous"),
                          list(name = "SEX", use = "covariate", type = "categorical"),
                          list(name = "WEIGHT", use = "covariate", type = "continuous"),
                          list(name = "BMI", use = "covariate", type = "continuous"))

theta <- tribble(~NAME, ~TRANS, ~INIT, ~LB, ~UB, ~EST,
                  "Cl", "logNormal", 0.2, NA, NA, TRUE,
                "V", "logNormal", 20, NA, NA, TRUE,
                "ka", "logNormal", 0.2, NA, NA, TRUE
)

ruv <- list(YNAME = "y1", DVID = 1, TRANS = "normal", PRED = "Cc",
           ERR = "combined1", INIT = c(1, 1), EST = c(TRUE, TRUE), BLQM = NULL)

re <- list(init = tribble(~Cl, ~V, ~ka,
                           1, 0, 0,
                           0, 1, 0,
                           0, 0, 1) %>% as.matrix(),
            est = tribble(~Cl, ~V, ~ka,
                         TRUE, NA, NA,
                           NA, TRUE, NA,
                          NA, NA, TRUE) %>% as.matrix())
occ <- list(init = tribble(~Cl, ~V, ~ka,
                        0, 0, 0,
                          0, 0, 0,
                          0, 0, 0) %>% as.matrix(),
           est = tribble(~Cl, ~V, ~ka,
                         NA, NA, NA,
                         NA, NA, NA,
                         NA, NA, NA) %>% as.matrix())
covs <- list(list(PAR = "V", COVNAME = "AGE", FUNC = "linear",
                 TRANS = "median", INIT = 1, EST = TRUE),
             list(PAR = "ka", COVNAME = "SEX", REF = 0, INIT = 1, EST = TRUE))

gco <-list(headers = headers, data = data, model = model, task_opt = "",
          covs = covs, project_name = "test-proj", theta = theta,
          ruv = ruv, re = re, occ = occ, modelText = "")

res <- sg_vpc_sim(fpath_i, gco = gco, output = "Cc")
```

`sg_vpc_vis` *Visual Predictive Check (VPC) Function*

Description

Generates VPC plots to assess model performance by comparing observed data with prediction intervals derived from simulated data.

Usage

```
sg_vpc_vis(
  ds_sim,
  data_i,
  output_names,
  time_col = "TIME",
  dv_col = "DV",
  log_y = FALSE,
  piLow = 0.1,
  piUp = 0.9,
  ciLow = 0.025,
  ciUp = 0.975,
  pred.corr = FALSE,
  lab_y = "DV",
  lab_x = "Time, h",
  theor_perc = TRUE,
  theor_percCI = TRUE,
  emp_perc = TRUE,
  dt_obs_fl = FALSE,
  legend_fl = FALSE,
  n_bins = 10,
  method = "kmeans",
  interpolation = TRUE,
  strat_by_dose = NULL
)
```

Arguments

`ds_sim`	GSO. Data frame with simulated data containing columns: • ID: simulation replicate identifier • TIME: time points • VAR: output variable name • VALUE: simulated values
`data_i`	A data frame with observed data containing columns: • ID: patient identifier • TIME: time points • DV: observed dependent variable values • DVID: dependent variable identifier
`output_names`	A data frame mapping `VAR` values (from `ds_sim`) to `DVID` values (from `data_i`). Must contain columns: • output: character, `VAR` values (e.g., "Cc", "Cp") • dvid: numeric, corresponding DVIDs (e.g., 1, 2)
`time_col`	String. The column to use as a time column. Currently, can be only `TIME`. Default is `TIME`.
`dv_col`	Character. Name of DV column in data_i. Default is `DV`

log_y	Logical. If TRUE, a logarithmic scale is applied to y-axis. Default is FALSE.
piLow	Numeric. Lower prediction interval bound. Default is 0.10.
piUp	Numeric. Upper prediction interval bound Default is 0.90.
ciLow	Numeric. Lower confidence interval bound. Default is 0.025
ciUp	Numeric. Upper confidence interval bound. Default is 0.975
pred.corr	Logical. Apply prediction correction. Default is FALSE.
lab_y	String. Y-axis label. Default is "DV".
lab_x	String. X-axis label. Default is "TIME, h".
theor_perc	Logical. Show theoretical percentiles. Default is TRUE.
theor_percCI	Logical. Show CI around theoretical percentiles. Default is TRUE.
emp_perc	Logical. Show empirical percentiles. Default is TRUE
dt_obs_fl	Logical. Show observed data points. Default is FALSE
legend_fl	Logical. Show legend. Default is FALSE.
n_bins	Integer. Number of bins to use in the histogram. Default is 10.
method	Character. Binning method: "kmeans", "equal", "quantile" (default: "kmeans").
interpolation	Logical. Use line interpolation vs rectangles (default: FALSE).
strat_by_dose	Character. Variable name for dose stratification (default: NULL).

Details

For now, the model in the example is NOT a GMO, as the parameters in generalized fit object are backtransformed, and, therefore, not transformed in the model. This issue will be fixed in the next versions of SimuRg package

Value

List of ggplot objects, one for each output variable

Examples

```
library(rxode2)
fpath_i <- system.file("extdata", "simurg_object", "Warfarin_PK.RData", package = "SimuRg")
mod <- rxode2::rxode2({
  ka_pop = 0.1;
  Vd_pop = 10;
  Cl_pop = 0.5;

  omega_ka = 0;
  omega_Vd = 0;
  omega_Cl = 0;

  Cc_b = 0;
  ka_tv = exp(log(ka_pop));
  Vd_tv = exp(log(Vd_pop));
  Cl_tv = exp(log(Cl_pop));
```

```
  ka = ka_tv * exp(omega_ka);
  Vd = Vd_tv * exp(omega_Vd);
  Cl = Cl_tv * exp(omega_Cl);

  Cc = Ac / Vd;

  Ad(0) = 0;
  Ac(0) = 0;

  d/dt(Ad) = -ka * Ad;
  d/dt(Ac) = ka * Ad - Cl * Cc;

  Cc_ResErr = Cc * (1 + Cc_b);
})

sim_data <- sg_vpc_sim(fpath_i, model=mod, outputs = "Cc_ResErr")
outp_nms <- data.frame(dvid = 1, output = "Cc")
vpc_plots <- sg_vpc_vis(
  ds_sim = sim_data,
  data_i = warfarin,
  output_names = outp_nms,
  lab_x = "Time (hours)",
  lab_y = "Concentration (ng/mL)",
  n_bins = 8,
  pred.corr = TRUE
)
vpc_plots[[1]]
```

`sg_vpop_est` *Perform generation of synthetic datasets for an empirical distribution*

Description

The function operates in two modes:

- **Fixed seed mode** (when `seed` is specified): Generates a single dataset using the provided seed
- **Search mode** (when `seed = NA`): Iteratively searches for `nds` datasets that meet the correlation difference threshold

Usage

```
sg_vpop_est(
  data_i,
  nobj = NA,
  id_col = NULL,
  minnumlev = 3,
  excl_col = NULL,
```

```
  seed = 123,
  seed_umap = 42,
  palette = NULL,
  diag_plots = FALSE,
  remove_duplicates = TRUE,
  noise_level = 0.1,
  nds = 1,
  tg_corrdif = 0.1
)
```

Arguments

data_i	data frame. Input data frame containing the original dataset to be synthesized (required)
nobj	integer. Specify the exact number of rows to generate in the synthetic dataset. When NA, uses the original dataset size (optional, default: NA)
id_col	Character string. Specify the name of the identifier column to exclude from synthesis. Default: NULL.
minnumlev	integer. Threshold; numeric variables with <= minnumlev unique values are converted to factors (optional, default: 3)
excl_col	Character vector. Contains column names to exclude from synthesis. Default: NULL
seed	integer. Random seed for synthetic data generation. If provided (not NA, not NULL), generates a single dataset with this seed (fixed seed mode). If NA or NULL uses search mode to find nds datasets meeting correlation threshold (optional, default: 123)
seed_umap	integer. Random seed for UMAP algorithm reproducibility (optional, default: 42)
palette	character vector. Contains color codes (hex format) for custom plot color schemes. If provided, should contain at least 2 colors. Used for histograms, bar plots, and UMAP visualizations (optional, default: c("#3a6eba", "#efdd3c", "#1a1866", "#f2b93b"))
diag_plots	logical flag. If TRUE, generates diagnostic plots and UMAP visualizations (optional, default: TRUE)
remove_duplicates	logical flag. If TRUE, automatically removes exact duplicates between original and synthetic data by adding controlled noise (optional, default: TRUE)
noise_level	numeric. Proportion of standard deviation to use when adding noise to continuous variables for duplicate removal. E.g., 0.10 means 10% of SD (optional, default: 0.10)
nds	integer. Number of synthetic datasets to generate in search mode. Ignored in fixed seed mode (optional, default: 1)
tg_corrdif	numeric. Target maximum absolute correlation difference threshold for dataset selection in search mode. Ignored in fixed seed mode (optional, default: 0.1)

Value

Output format depends on mode:

- In fixed seed mode (when seed is specified), returns a single list with:
  - datagen - Synthetic dataset
  - seed - Random seed used to generate this dataset (integer or NA)
  - exact_dupl_check - Logical indicating if exact duplicates exist between original and synthetic data
  - dplot_umap - ggplot object with combined UMAP visualization comparing original and synthetic data (or NULL if diag_plots=FALSE)
  - ks_test - Tibble with Kolmogorov-Smirnov results for continuous variables (variable, formatted p.value, status, d_statistic); when ties are present, a warning is emitted that the test may be approximate (or NULL if no continuous variables)
  - jsd_res - Tibble with per-variable Jensen-Shannon divergence (JSD) for categorical variables (variable, jsd, n_levels), with weighted mean stored in attribute jsd_weighted_mean (or NULL if no categorical variables)
  - corr_diff_mean - Mean absolute difference between original and synthetic correlation matrices (numeric or NULL if no continuous variables)
  - corr_diff_max - Maximum absolute difference between original and synthetic correlation matrices (numeric or NULL if no continuous variables)
  - dplot_corr_diff - ggplot heatmap object showing correlation difference matrix (Synthetic - Original) (or NULL if diag_plots=FALSE or no continuous variables)
  - dplot_cont - List of ggplot histograms for continuous variables (or NULL if diag_plots=FALSE or no continuous variables)
  - dplot_cat - List of ggplot barplots for categorical variables (or NULL if diag_plots=FALSE or no categorical variables)
- In search mode (when seed = NA), returns a list of such lists, one per generated dataset.

Examples

```
library(dplyr)

# Generate example dataset
data <- data.frame(
  id = 1:150,
  ALB = rnorm(150, mean = 3.5, sd = 0.5),
  ALT = rnorm(150, mean = 50, sd = 20),
  SEX = factor(sample(c("M", "F"), 150, replace = TRUE)),
  RACE = factor(sample(c("White", "Black", "Asian", "Other"), 150, replace = TRUE))
)

# EXAMPLE 1: Fixed seed mode - generate single dataset with specified seed
output_fixed <- sg_vpop_est(data_i = data,
                            id_col = "id",
                            seed = 123,        # Fixed seed mode
                            seed_umap = 40,
                            diag_plots = TRUE)
```

```
# Access the dataset and its diagnostics
print(head(output_fixed[[1]]$datagen, 10))
print(output_fixed[[1]]$ks_test)
print(output_fixed[[1]]$seed)           # Will be 123
print(output_fixed[[1]]$corr_diff_max)
print(output_fixed[[1]]$dplot_umap)

# EXAMPLE 2: Search mode - generate 3 datasets meeting correlation threshold
output_search <- sg_vpop_est(data_i = data,
                             id_col = "id",
                             seed = NA,        # Search mode (default)
                             seed_umap = 40,
                             diag_plots = TRUE,
                             nds = 3,          # Generate 3 datasets
                             tg_corrdif = 0.1) # Target correlation difference

# Compare metrics across all datasets
sapply(output_search, function(x) x$corr_diff_max)
sapply(output_search, function(x) x$jsd_res)
sapply(output_search, function(x) x$seed)  # Different seeds for each dataset
```

`warfarin` *Warfarin PKPD dataset*

Description

Generated dataset with warfarin PK/PD measurements and covariates for 100 patients.

Usage

```
warfarin
```

Format

`warfarin`**:**

A data frame with 1700 rows and 17 columns:

ID identifier of the individual

TIME time of the dose or observation record (nominal)

DV records the measurement data

DVID identifier for the observation type (to distinguish different types of observations, e.g PK and PD)

DVNAME name for the observation type (to distinguish different types of observations, e.g PK and PD)

CMT compartment of the event

ADM identifier for the type of dose

AMT dose amount
EVID identifier to indicate if the line is a dose-line or a response-line
MDV identifier to ignore the observation information of that line
AGE age of the individual
SEX sex of the individual
WEIGHT weight of the individual
BMI BMI of the individual
CLCR creatinine clearance of the individual
CYP2C9_gentyp CYP2C9 genotype of the individual
VKORC1_gentyp VKORC1 genotype of the individual

Index