Package ‘mrgsim.sa’

December 8, 2023

Type Package
Title Sensitivity Analysis with 'mrgsolve'
Version 0.2.0
Maintainer Kyle Baron <kylebtwin@imap.cc>
Description Perform sensitivity analysis on ordinary differential equation based models, including ad-hoc graphical analyses based on structured sequences of parameters as well as local sensitivity analysis. Functions are provided for creating inputs, simulating scenarios and plotting outputs.
License GPL (>= 2)
URL https://github.com/kylebaron/mrgsim.sa
BugReports https://github.com/kylebaron/mrgsim.sa/issues
Suggests testthat, knitr, rmarkdown
Imports withr, purrr, dplyr, assertthat, rlang, ggplot2, tidyselect, tidyr, methods, tibble, patchwork, glue
Encoding UTF-8
Language en-US
Depends mrgsolve
RoxygenNote 7.2.3
VignetteBuilder knitr
Collate 'utils.R' 'parseq.R' 'sens.R' 'AAA.R' 'lsa.R' 'sens_each.R'
  'sens_grid.R' 'sens_plot.R' 'sens_run.R' 'seq.R'
NeedsCompilation no
Author Kyle Baron [aut, cre]
Repository CRAN
Date/Publication 2023-12-08 13:50:02 UTC
R topics documented:

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>lsa</td>
<td>2</td>
</tr>
<tr>
<td>mrgsim.sa</td>
<td>3</td>
</tr>
<tr>
<td>parseq_cv</td>
<td>4</td>
</tr>
<tr>
<td>parseq_fct</td>
<td>5</td>
</tr>
<tr>
<td>parseq_manual</td>
<td>6</td>
</tr>
<tr>
<td>parseq_range</td>
<td>6</td>
</tr>
<tr>
<td>parseq_reference</td>
<td>7</td>
</tr>
<tr>
<td>select_par</td>
<td>8</td>
</tr>
<tr>
<td>select_sens</td>
<td>8</td>
</tr>
<tr>
<td>sens_fun</td>
<td>9</td>
</tr>
<tr>
<td>sens_plot</td>
<td>10</td>
</tr>
<tr>
<td>sens_run</td>
<td>12</td>
</tr>
<tr>
<td>seq_cv</td>
<td>13</td>
</tr>
<tr>
<td>seq_even</td>
<td>14</td>
</tr>
<tr>
<td>seq_fct</td>
<td>14</td>
</tr>
<tr>
<td>seq_geo</td>
<td>15</td>
</tr>
</tbody>
</table>

Index

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
</tr>
</tbody>
</table>

lsa

Perform local sensitivity analysis

Description

Perform local sensitivity analysis

Usage

```
lsa(mod, par, var, fun = .lsa_fun, eps = 1e-07, ...)
```

```
lsa_plot(x, ...)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mod</td>
<td>a mrgsolve model object.</td>
</tr>
<tr>
<td>par</td>
<td>parameter names as character vector or comma-separated string.</td>
</tr>
<tr>
<td>var</td>
<td>output names (compartment or capture) as character vector or comma-separated string.</td>
</tr>
<tr>
<td>fun</td>
<td>generating simulated for sensitivity analysis (see details).</td>
</tr>
<tr>
<td>eps</td>
<td>parameter change value for sensitivity analysis.</td>
</tr>
<tr>
<td>...</td>
<td>passed to plot.lsa().</td>
</tr>
<tr>
<td>x</td>
<td>output from lsa().</td>
</tr>
</tbody>
</table>
Value

A tibble with class lsa.

Examples

```r
mod <- mrgsolve::house(delta=0.1)
par <- "CL,VC,KA"
var <- "CP"
dose <- ev(amt = 100)
fun <- function(mod, ...) mrgsolve::mrgsim_e(mod, dose, output="df")
out <- lsa(mod, par, var, fun)
head(out)
lsa_plot(out)
```

Description

Perform local sensitivity analysis on ordinary differential equation based models, including ad-hoc graphical analyses based on sequences of parameters as well as local sensitivity analysis. Functions are provided for creating inputs, simulating scenarios and plotting outputs.

Details

- Local sensitivity analysis: `lsa()`
- Run ad-hoc sensitivity analyses: `sens_each()`, `sens_grid()`, `sens_run()`
  - Use `sens_each_data()` and `sens_grid_data()` to pass in data sets
- Parameter sequence generation:
  - In a pipeline: `parseq_cv()`, `parseq_fct()`, `parseq_range()`, `parseq_manual()`
  - Stand alone: `seq_cv()`, `seq_fct()`, `seq_geo()`, `seq_even()`
- Plot ad-hoc sensitivity analysis results
  - Use `sens_plot()`
- Select a subset of sensitivity analysis results
  - Use `select_sens()`
parseq_cv

Generate a sequence of parameters based on CV

Description

Generate a sequence of parameters based on CV

Usage

parseq_cv(mod, ..., .cv = 30, .n = 5, .nsd = 2, .digits = NULL)

Arguments

mod a model object.
... model parameter names.
.cv a coefficient of variation used to determine range of test parameters.
.n number of parameters to simulate in the sequence.
.nsd number of standard deviations used to determine the range.
.digits if numeric, the number of significant digits in the parameter sensitivity values are set using base::signif().

Details

• .cv is passed to seq_cv() as cv
• .n is passed to seq_cv() as n
• .nsd is passed to seq_cv() as nsd

See Also

parseq_fct(), parseq_range(), parseq_manual()

Examples

mod <- mrgsolve::house()

mod %>%
  parseq_cv(CL, VC) %>%
sens_each()
parseq_fct  Generate a sequence of parameters

Description
Generate a sequence of parameters

Usage
parseq_fct(mod, ..., .n = 5, .factor = 2, .geo = TRUE, .digits = NULL)

parseq_factor(mod, ..., .n = 5, .factor = 2, .geo = TRUE, .digits = NULL)

Arguments
mod  a model object.
... unquoted parameter names.
.n  number of parameters to simulate between the minimum and maximum parameter values.
.factor  a numeric vector used to divide and multiply the parameter value thus generating the minimum and maximum parameter values, respectively, for the sequence; if .factor is length 1 it will be recycled to length 2: the first value is used to divide the nominal value generating the minimum value; the second value is used to multiply the nominal value generating the maximum value.
.geo  if TRUE a geometric sequence is generated (evenly spaced from min to max on log scale); otherwise, the sequence is evenly spaced on Cartesian scale.
.digits  if numeric, the number of significant digits in the parameter sensitivity values are set using base::signif().

Details
• .n is passed to seq_fct() as n
• .factor is passed to seq_fct() as factor

See Also
parseq_cv(), parseq_range(), parseq_manual()

Examples
mod <- mrgsolve::house()

mod %>%
  parseq_fct(CL, VC) %>%
  sens_each()
parseq_manual

Simulation helper to manually specify parameter sequences

Description
Simulation helper to manually specify parameter sequences

Usage
parseq_manual(mod, ...)

Arguments
mod          mrgsolve model object.
...          named numeric vectors of parameter values to simulate; names must correspond
to parameters in the model object.

Details
Parameter value vectors passed via ... will be sorted prior to simulation.

See Also
parseq_cv(), parseq_range(), parseq_fct()

Examples
mod <- mrgsolve::house()

mod %>%
   parseq_manual(CL = c(0.5, 1, 1.5)) %>%
   sens_each()

parseq_range

Simulation helper to generate a sequence of parameters from a range

Description
Simulation helper to generate a sequence of parameters from a range

Usage
parseq_range(mod, ..., .n = 5, .geo = TRUE, .digits = NULL)
Arguments

mod  mrgsolve model object.
... named parameter range vectors (minimum and maximum) for model parameters; each vector must have length 2 and names must correspond to model parameters.
.n  number of values to simulate for each parameter sequence; passed to seq_geo() as n.
.geo  if TRUE generate a geometric sequence; otherwise, generate a sequence evenly spaced on Cartesian scale; see seq_geo().
.digits  if numeric, the number of significant digits in the parameter sensitivity values are set using base::signif().

Details

Parameter range vectors passed via ... will be sorted prior to simulation.

See Also

parseq_cv(), parseq_fct(), parseq_manual()

Examples

mod <- mrgsolve::house()

mod %>%
  parseq_range(CL = c(0.5,1),VC = c(10,40)) %>%
  sens_each()
select_par

Identify parameters in a model for sensitivity analysis

Description
Identify parameters in a model for sensitivity analysis

Usage
select_par(mod, ...)

Arguments
mod       an mrgsolve model object.
...       unquoted parameter names.

Examples
mod <- mrgsolve::house()
select_par(mod, CL, VC)

select_sens

Select sensitivity runs from a sens_each object

Description
Select sensitivity runs from a sens_each object

Usage
select_sens(x, dv_name = NULL, p_name = NULL)

Arguments
x             a sens_each object.
dv_name       character names of dependent variables to select; can be a comma-separated string.
p_name        character names of parameters to select; can be a comma-separated string.

Value
The updated sens_each object is returned.
Examples

```r
library(dplyr)
mod <- mrgsolve::house()
out1 <- mod %>% parseq_factor(CL,VC) %>% sens_each()
out2 <- select_sens(out1, dv_name = "CP,RESP", p_name = "CL")
```

---

**sens_fun**

**Run an ad-hoc sensitivity analysis**

**Description**

Use `sens_each()` to examine sequences of parameters, one at a time. Use `sens_grid()` to examine all combinations of sequences of parameters. The `sens_each_data()` and `sens_grid_data()` variants allow you to pass in a data set to simulate from.

**Usage**

```r
sens_each(mod, idata = NULL, ...)
sens_each_data(mod, data, idata = NULL, ...)
sens_grid(mod, idata = NULL, ...)
sens_grid_data(mod, data, idata = NULL, ...)
```

**Arguments**

- `mod`: an mrgsolve model object (usually read in with `mrgsolve::mread()`).
- `idata`: included only to prevent users from passing through; the function will create an `idata_set` if appropriate.
- `...`: passed to `mrgsolve::mrgsim_d()`.
- `data`: a simulation input data set (see `mrgsolve::data_set()`).

**Value**

A tibble-like object with class `sens_each` or `sens_grid`, depending on the vary method that was used. These objects will look just like a tibble, but they can be plotted with `sens_plot()`.

**See Also**

`sens_plot()`
Examples

```r
mod <- mrgsolve::house()

mod <- mrgsolve::ev(mod, amt = 100)

out_each <- parseq_cv(mod, CL, VC, .n = 3) %>% sens_each()

sens_plot(out_each, dv_name = "CP,RESP", layout = "facet_grid")

out_grid <- parseq_cv(mod, CL, VC) %>% sens_grid()

sens_plot(out_grid, dv_name = "CP")
```

sens_plot  
*Plot sensitivity analysis results*

Description

Plot sensitivity analysis results

Usage

```r
sens_plot(data, ...)

## S3 method for class 'sens_each'
sens_plot(
  data,
  dv_name = NULL,
  p_name = NULL,
  logy = FALSE,
  ncol = NULL,
  lwd = 0.8,
  digits = 3,
  plot_ref = TRUE,
  xlab = "time",
  ylab = dv_name[1],
  layout = c("default", "facet_grid", "facet_wrap", "list"),
  grid = FALSE,
  ...
)

## S3 method for class 'sens_grid'
sens_plot(
  data,
  dv_name = NULL,
  digits = 2,
```
sens_plot

ncol = NULL,
lwd = 0.8,
logy = FALSE,
plot_ref = TRUE,
...)
Arguments
data output from sens_each() or sens_grid().
... arguments passed on to methods.
dv_name dependent variable names to plot; can be a comma-separated string; if NULL, then the unique values of dv_name in data are used.
p_name parameter names to plot; can be a comma-separated string.
logy if TRUE, y-axis is transformed to log scale
ncol passed to ggplot2::facet_wrap().
lwd passed to ggplot2::geom_line().
digits used to format numbers on the strips.
plot_ref if TRUE, then the reference case will be plotted in a black dashed line.
xlab x-axis title.
ylab y-axis title; not used for facet_grid or facet_wrap layouts.
layout specifies how plots should be returned when dv_name requests multiple dependent variables; see Details.
grid if TRUE, plots from the sens_each method will be arranged on a page with patchwork::wrap_plots(); see the ncol argument.

Details
The layout argument is only used for the sens_each method. It lets you get the plots back in different formats when multiple dependent variables are requested via dv_name.

- Use default to get the plots back in a list if multiple dependent variables are requested otherwise a single plot is returned.
- Use facet_grid to get a single plot, with parameters in columns and dependent variables in rows.
- Use facet_wrap to get a plot with faceted using ggplot2::facet_wrap(), with both the parameter name and the dependent variable name in the strip.
- Use list to force output to be a list of plots; this output can be further arranged using patchwork::wrap_plots() if desired.

When grid is TRUE, a list of plots will be returned when multiple dependent variables are requested.

Value
A ggplot object when one dv_name is specified or a list of ggplot objects when multiple dv_names are specified.
Examples

```r
mod <- mrgsolve::house()

dose <- mrgsolve::ev(amt = 100)

out <- sens_run(mod, sargs = list(events = dose), par = "CL,VC")
sens_plot(out, dv_name = "CP")
```

Description

Run ad-hoc parameter sensitivity analyses with mrgsolve

Usage

```r
sens_run(
  mod,
  par = NULL,
  var = NULL,
  method = c("factor", "cv", "range", "manual"),
  vary = c("each", "grid"),
  ...,
  sargs = list()
)
```

Arguments

- **mod**: a mrgsolve model object.
- **par**: parameter names for sensitivity analysis; this can be a character vector or a comma-separated string (see examples).
- **var**: names of model output variables to include in simulated output; this could be the name of a compartment or another output derived inside of the model (e.g. DV or CP or logV, but is specific to what is coded into `mod`).
- **method**: parameter sequence generation method.
- **vary**: use each to vary one parameter at a time or grid to vary all combinations of parameters.
- **...**: passed to method function.
- **sargs**: a named list of arguments passed to `sens_each()` or `sens_grid()` and eventually to `mrgsolve::mrgsim()`.
Examples

```r
mod <- mrgsolve::house()

dose <- mrgsolve::ev(amt = 100)

sens_run(
  mod,
  par = "CL,VC",
  method = "cv",
  vary = "each",
  sargs = list(events = dose)
)
```

seq_cv

Generate a sequence based on coefficient of variation

Description

Generate a sequence based on coefficient of variation

Usage

```r
seq_cv(point, cv = 30, n = 5, nsd = 2, digits = NULL)
```

Arguments

- **point**: reference parameter value.
- **cv**: coefficient of variation.
- **n**: number of values to simulate in the sequence.
- **nsd**: number of standard deviations defining the range of simulated values.
- **digits**: number of significant digits in the answer; if NULL (the default) all digits are retained.

Examples

```r
seq_cv(10)
seq_cv(5, n = 10)
```
seq_even

Generate evenly spaced sequence

Description

Generate evenly spaced sequence

Usage

seq_even(from, to, n = 5, digits = NULL)

Arguments

from passed to base::seq().
to passed to base::seq().
n passed to base::seq() as length.out.
digits number of significant digits in the answer; if NULL (the default) all digits are retained.

Examples

seq_even(1, 10, 4)

seq_fct

Generate a sequence by fold increase and decrease from a point

Description

Generate a sequence by fold increase and decrease from a point

Usage

seq_fct(point, n = 5, factor = c(3, 3), geo = TRUE, digits = NULL)

Arguments

point a numeric vector of length 1.
n number of elements in the sequence.
factor an integer vector of length 1 or 2; if length 1, values will be recycled to length 2; the first number used to divide point to generate the minimum value in the sequence; the second number is used to multiply point to generate the maximum value in the sequence.
geo if TRUE, seq_geo() is used to generate the sequence; otherwise, seq_even() is used to generate the sequence.
digits number of significant digits in the answer; if NULL (the default) all digits are retained.
seq_geo

Examples

seq_fct(10)
seq_fct(10, n = 4, factor = 2)
seq_fct(10, n = 4, factor = 2, geo = TRUE)

seq_geo

Generate a geometric sequence of parameter values

Description

Generate a geometric sequence of parameter values

Usage

seq_geo(from, to, n = 5, digits = NULL)

Arguments

from passed to base::seq(); must be numeric and positive.
to passed to base::seq(); must be numeric and positive.
n passed to base::seq() as length.out.
digits number of significant digits in the answer; if NULL (the default) all digits are retained.

Examples

seq_geo(from = 1, to = 10, n = 10)
Index

base::seq(), 14, 15
base::signif(), 4, 5, 7

ggplot2::facet_wrap(), 11
ggplot2::geom_line(), 11

lsa, 2
lsa(), 3
lsa_plot(lsa), 2

mrgsim.sa, 3
mrgsolve::data_set(), 9
mrgsolve::mread(), 9
mrgsolve::mrgsim(), 12
mrgsolve::mrgsim_d(), 9

parseq_cv, 4
parseq_cv(), 3, 5–7
parseq_factor (parseq_fct), 5
parseq_fct, 5
parseq_fct(), 3, 4, 6, 7
parseq_manual, 6
parseq_manual(), 3–5, 7
parseq_range, 6
parseq_range(), 3–6
parseq_reference, 7
patchwork::wrap_plots(), 11
plot.lsa(), 2

select_par, 8
select_sens, 8
select_sens(), 3
sens_each (sens_fun), 9
sens_each(), 3, 11, 12
sens_each_data (sens_fun), 9
sens_each_data(), 3
sens_fun, 9
sens_grid (sens_fun), 9
sens_grid(), 3, 11, 12
sens_grid_data (sens_fun), 9
sens_grid_data(), 3

sens_plot, 10
sens_plot(), 3, 9
sens_run, 12
sens_run(), 3
seq_cv, 13
seq_cv(), 3, 4
seq_even, 14
seq_even(), 3, 14
seq_fct, 14
seq_fct(), 3, 5
seq_geo, 15
seq_geo(), 3, 7, 14