Package: ospsuite.globalsensitivity (via r-universe)

July 15, 2026

Type Package

Title Open Systems Pharmacology Global Sensitivity Package

Version 1.0.0

Author Abdullah Hamadeh (Systems In Silico Ltd.)

Maintainer Abdullah Hamadeh <ahamadeh@systemsinsilico.com>

Description A global sensitivity analysis for Open Systems Pharmacology models.

License GPL-2

URL https://github.com/Open-Systems-Pharmacology/OSPSuite.GlobalSensitivity, https://www.open-systems-pharmacology.org/OSPSuite.GlobalSensitivity/

BugReports https://github.com/Open-Systems-Pharmacology/OSPSuite.GlobalSensitivity/issues

Depends ospsuite, ggplot2, shiny, shinyTree, shinyjs, dplyr, parallel, randtoolbox, writexl, stats, tictoc

Imports interp, utils

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

Remotes ospsuite=Open-Systems-Pharmacology/OSPSuite-R@*release, rSharp=Open-Systems-Pharmacology/rSharp@*release, ospsuite.utils=Open-Systems-Pharmacology/OSPSuite.RUtils@*release, tlf=Open-Systems-Pharmacology/TLF-Library@*release, ospsuite.plots=Open-Systems-Pharmacology/OSPSuite.Plots@*release

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake dotnet-runtime-8.0 make libicu-dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev libssl-dev libx11-dev zlib1g-dev

Repository https://open-systems-pharmacology.r-universe.dev

Date/Publication 2026-07-03 08:53:55 UTC

RemoteUrl https://github.com/Open-Systems-Pharmacology/OSPSuite.GlobalSensitivity

RemoteRef v1.0.0

RemoteSha 61bddd50b2b799bfabdf3cfea7cbbe20f6636609

Contents

`distribution` *distribution*

Description

A named list of distribution constructors used to create distribution objects for global sensitivity analyses. Each element is a constructor function returning a distribution object: `Uniform`, `LogUniform`, `Normal` and `LogNormal`.

Usage

```
distribution
```

Format

A named `list` of distribution constructor functions.

generateEFASTBarGraph *generateEFASTBarGraph*

Description

Function to generate a bar graph of EFAST sensitivity analysis results.

Usage

```
generateEFASTBarGraph(gsaResultsDataframe)
```

Arguments

gsaResultsDataframe	EFAST results returned by the runEFAST function.

Value

A list of ggplot bar graphs, one corresponding to each output path/PK parameter combination.

generateMorrisPlot *generateMorrisPlot*

Description

Function to generate a plot of Morris sensitivity analysis.

Usage

```
generateMorrisPlot(morrisResults, logPlot = FALSE)
```

Arguments

morrisResults	Morris sensitivity results returned by runMorris function.
logPlot	Logical setting. The Morris results are plotted on a logarithmic scale if TRUE.

Value

A list of ggplots of Morris sensitivity analysis results, one corresponding to each output path/PK parameter combination.

generateSobolBarGraph *generateSobolBarGraph*

Description

Function to generate a bar graph of Sobol sensitivity analysis results.

Usage

```
generateSobolBarGraph(gsaResultsDataframe)
```

Arguments

gsaResultsDataframe	Sobol results returned by the runSobol function.

Value

A list of ggplot bar graphs, one corresponding to each output path/PK parameter combination.

generateTornadoPlot *generateTornadoPlot*

Description

Function to generate a tornado plot of local sensitivity analysis or uncertainty analysis results.

Usage

```
generateTornadoPlot(
  sensitivityDataFrame,
  generateForUncertaintyAnalysis = FALSE
)
```

Arguments

sensitivityDataFrame	Sensitivity or uncertainty analysis results returned by runSU function.
generateForUncertaintyAnalysis	Logical value. If TRUE, the tornado plots will be generated for uncertainty analysis results.

Value

A list of ggplot tornado plots results, one corresponding to each output path/PK parameter combination.

getContourPlot *Generate Response Surface Contour Plots for eFAST Results*

Description

Generates a matrix of contour plots visualizing the pairwise response surfaces of model outputs based on eFAST sensitivity analysis results.

Usage

```
getContourPlot(efastResults, jitterSize = 0, gridSize = 40, logScale = TRUE)
```

Arguments

`efastResults`	A list object containing the complete results of an eFAST sensitivity analysis. Expected structure: **InputOutputDf** Data frame containing simulation inputs, output paths, PK parameters, and calculated values. **Parameters** Data frame or list containing parameter metadata, specifically `path` and `displayName`. **Results** Data frame containing calculated sensitivity indices (`Measure`, `Value`, `ParameterDisplayName`). **Outputs** Data frame or list containing output metadata (`path`, `displayName`).
`jitterSize`	A numeric value representing the noise magnitude added during interpolation. Passed to `getPairwiseGrid`. Defaults to 0 (though `getPairwiseGrid` may apply a default if needed).
`gridSize`	An integer specifying the resolution of the contour grids. Defaults to 40.
`logScale`	Logical. If `TRUE` (default), the response variable (Z-axis) is log-transformed (log10) before plotting. This is recommended for PK data with large dynamic ranges. If data contains zeros or negative values, the function automatically falls back to a linear scale with a warning.

Details

This function creates a visualization tool to identify parameter interactions and non-linearities. The process involves:

1. Iterating through every unique Output and PK parameter (e.g., "C_max", "AUC") in the results.
2. For each Output/PK combination, extracting the relevant simulation data.
3. Calling `getPairwiseGrid` to generate interpolated surfaces for all parameter pairs.
4. Optionally applying a log-transformation to the response values (Z-axis) to handle wide dynamic ranges common in PK/PD.
5. Ranking parameters based on their Total Sensitivity index (S_T) so that the most influential parameters appear in the top-left of the plot matrix.

6. Constructing a ggplot2 object using geom_contour_filled and facet_grid to display the full matrix of pairwise interactions.

The resulting plot is a scatterplot matrix where:

- Off-diagonal cells display filled contours of the response surface for two parameters.
- Diagonal cells display the names of the parameters.

Value

A nested list of ggplot objects, structured as: plotList[[outputName]][[pkParameter]]. Each element is a complete ggplot object ready for printing or saving.

LogNormalDistribution *LogNormalDistribution*

Description

R6 class, child class of SADistribution, defining a statistical lognormal distribution object

Super class

ospsuite.globalsensitivity::SADistribution -> LogNormalDistribution

Active bindings

type is the type of the probability distribution.

mean The mean of the lognormal distribution.

CV The coefficient of variation of the lognormal distribution.

Methods

Public methods:

- LogNormalDistribution$new()
- LogNormalDistribution$quantilesToSample()
- LogNormalDistribution$clone()

Method new(): Create a new LogNormalDistribution object.

Usage:

```
LogNormalDistribution$new(mean, CV)
```

Arguments:

mean The mean of the lognormal distribution.

CV The coefficient of variation of the lognormal distribution.

Returns: An instance of the LogNormalDistribution class.

Method `quantilesToSample()`**:** Maps a vector of `quantiles` to corresponding values from the lognormal distribution.

Usage:

```
LogNormalDistribution$quantilesToSample(quantiles)
```

Arguments:

`quantiles` Vector of quantiles.

Returns: A vector of values from the lognormal distribution corresponding to the input `quantiles`.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
LogNormalDistribution$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`LogUniformDistribution`

LogUniformDistribution

Description

R6 class, child class of `SADistribution`, defining a statistical loguniform distribution object

Super class

`ospsuite.globalsensitivity::SADistribution` -> `LogUniformDistribution`

Active bindings

`type` is the type of the probability distribution.

`minimum` The maximum of the loguniform distribution.

`maximum` The maximum of the loguniform distribution.

Methods

Public methods:

- `LogUniformDistribution$new()`
- `LogUniformDistribution$quantilesToSample()`
- `LogUniformDistribution$clone()`

Method `new()`**:** Create a new `LogUniformDistribution` object.

Usage:

```
LogUniformDistribution$new(minimum, maximum)
```

Arguments:

`minimum` The minimum of the loguniform distribution.

`maximum` The maximum of the loguniform distribution.

Returns: An instance of the `LogUniformDistribution` class.

Method `quantilesToSample()`**:** Maps a vector of `quantiles` to corresponding values from the loguniform distribution.

Usage:

```
LogUniformDistribution$quantilesToSample(quantiles)
```

Arguments:

`quantiles` Vector of quantiles.

Returns: A vector of values from the loguniform distribution corresponding to the input `quantiles`.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
LogUniformDistribution$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`NormalDistribution` *NormalDistribution*

Description

R6 class, child class of `SADistribution`, defining a statistical normal distribution object

Super class

`ospsuite.globalsensitivity::SADistribution` -> `NormalDistribution`

Active bindings

`type` is the type of the probability distribution.

`mean` The mean of the normal distribution.

`stdv` The standard deviation of the normal distribution.

Methods

Public methods:

- `NormalDistribution$new()`
- `NormalDistribution$quantilesToSample()`
- `NormalDistribution$clone()`

Method `new()`**:** Create a new `NormalDistribution` object.

Usage:

```
NormalDistribution$new(mean, stdv)
```

Arguments:

`mean` The mean of the normal distribution.

`stdv` The standard deviation of the normal distribution.

Returns: An instance of the `NormalDistribution` class.

Method `quantilesToSample()`**:** Maps a vector of `quantiles` to corresponding values from the normal distribution.

Usage:

```
NormalDistribution$quantilesToSample(quantiles)
```

Arguments:

`quantiles` Vector of quantiles.

Returns: A vector of values from the normal distribution corresponding to the input `quantiles`.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
NormalDistribution$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`runEFAST` *runEFAST*

Description

Run the EFAST algorithm.

Usage

```
runEFAST(
  simulation,
  DDIsimulation = NULL,
  parameters,
  outputs,
  runParallel = TRUE,
  numberOfResamples = 1,
  updateProgress = NULL,
  saveResults = FALSE,
  saveFolder = NULL,
  saveFileName = NULL
)
```

Arguments

simulation	PKML simulation object.
DDIsimulation	DDI PKML simulation object.
parameters	List of SAParameter objects.
outputs	List of SAOutput objects.
runParallel	Logical value. EFAST sensitivity computation is run in parallel when TRUE.
numberOfResamples	Number of times to run the EFAST algorithm steps with resampling, as per Saltelli, Tarantola & Chan, 1999.
updateProgress	Logical value. Updates shiny app GUI with EFAST progress when TRUE.
saveResults	Logical value. Results are saved if TRUE.
saveFolder	Folder in which results will be saved if saveResults is set to TRUE.
saveFileName	File name to which results will be saved if saveResults is set to TRUE.

Value

Results of EFAST evaluation.

---

runGUI *runGUI*

Description

Function to run Shiny App GUI interface to ospsuite.globalsensitivity package.

Usage

```
runGUI()
```

---

runMorris *runMorris*

Description

Function to run Morris sensitivity analysis.

Usage

```
runMorris(
  simulation,
  DDIsimulation = NULL,
  parameters,
  outputs,
  numberOfSamples,
  runParallel = TRUE,
  updateProgress = NULL,
  saveResults = FALSE,
  saveFolder = NULL,
  saveFileName = NULL
)
```

Arguments

simulation	PKML simulation object.
DDIsimulation	DDI PKML simulation object.
parameters	List of SAParameter objects.
outputs	List of SAOutput objects.
numberOfSamples	The number of runs of the Morris algorithm to perform.
runParallel	Logical value. Morris computation is run in parallel when TRUE.
updateProgress	Logical value. Updates shiny app GUI with Morris algorithm progress when TRUE.
saveResults	Logical value. If TRUE, the results will be saved.
saveFolder	String indicating the path to the folder in which the results are to be saved.
saveFileName	String indicating the file name to use when saving the results.

Value

Morris sensitivity analysis results.

---

runSobol *runSobol*

Description

Function to run Sobol sensitivity analysis.

Usage

```
runSobol(
  simulation,
  DDIsimulation = NULL,
  parameters,
  outputs,
  numberOfSamples,
  runParallel = TRUE,
  updateProgress = NULL,
  saveResults = FALSE,
  saveFolder = NULL,
  saveFileName = NULL
)
```

Arguments

simulation	PKML simulation object.
DDIsimulation	DDI PKML simulation object.
parameters	List of SAParameter objects.
outputs	List of SAOutput objects.
numberOfSamples	The number of samples in parameter space at which to evaluate the simulation.
runParallel	Logical value. Sobol computation is run in parallel when TRUE.
updateProgress	Logical value. Updates shiny app GUI with Sobol algorithm progress when TRUE.
saveResults	Logical value. If TRUE, the results will be saved.
saveFolder	String indicating the path to the folder in which the results are to be saved.
saveFileName	String indicating the file name to use when saving the results.

Value

Sobol sensitivity analysis results.

---

runSU *runSU*

Description

Function to run sensitivity analysis and then run uncertainy analysis for parameters deemed sensitive. Main parameters:

Usage

```
runSU(
  simulation,
  DDIsimulation = NULL,
  customParameters = NULL,
  outputs,
  evaluateForAllParameters = FALSE,
  variationRange = 0.1,
  numberOfSensitivityAnalysisSteps = 2,
  sensitivityThreshold = 0.1,
  runUncertaintyAnalysis = TRUE,
  runUncertaintlyOnlyForSensitiveParameters = TRUE,
  numberOfUncertaintyAnalysisSamples = 100,
  quantiles = c(0.05, 0.5, 0.95),
  saveResults = FALSE,
  saveFolder = NULL,
  saveFileName = NULL,
  runParallel = TRUE,
  updateProgressSensitivity = NULL,
  updateProgressUncertainty = NULL
)
```

Arguments

simulation	PKML simulation object.
DDIsimulation	DDI PKML simulation object.
customParameters	List of SAParameter objects with user-specified distributions.
outputs	List of SAOutput objects.
evaluateForAllParameters	Logical value. If TRUE, all model parameters will be evaluated for sensitivity and those with a sensitivity greater than sensitivityThreshold will undergo uncertainty analysis. Any parameters not included in the customParameters list will have a log-uniform distribution with bounds set by variationRange. Sensitivity analysis settings:
variationRange	The variation range used to define the log-uniform distribution of the parameters in parameterPaths.
numberOfSensitivityAnalysisSteps	Positive integer. Number of steps at which the model is simulated within a parameter's variationRange during the sensitivity analysis.
sensitivityThreshold	Positive numeric value. The threshold sensitivity above which a parameter will undergo uncertainty analysis if runUncertaintyAnalysis and runUncertaintlyOnlyForSensitivePa are both TRUE. Uncertainty analysis settings:
runUncertaintyAnalysis	Logical value. If TRUE, uncertainty analysis will run.

runUncertaintlyOnlyForSensitiveParameters
: Logical value. If TRUE, uncertainty analysis will only be run for the parameters for which sensitivity exceeds sensitivityThreshold.

numberOfUncertaintyAnalysisSamples
: Positive integer value giving the number of Monte Carlo runs for each parameter.

quantiles
: Vector of numerical values between 0 and 1. The quantiles input sets the percentiles of the PK parameters to be evaluated in addition to the 50% and 95% percentiles, which are computed by default. Save settings:

saveResults
: Logical value. If TRUE, the results will be saved.

saveFolder
: String indicating the path to the folder in which the results are to be saved.

saveFileName
: String indicating the file name to use when saving the results. Parallel run and updating settings

runParallel
: Logical value. Sensitivity computation is run in parallel when TRUE.

updateProgressSensitivity
: Logical value. Updates shiny app GUI with sensitivity analysis progress when TRUE.

updateProgressUncertainty
: Logical value. Updates shiny app GUI with uncertainty analysis progress when TRUE.

Value

Sensitivity and uncertainty analysis results.

SAOutput *SAOutput*

Description

R6 class defining a model output object

Active bindings

path is the path to the output within the simulation PKML simulation.

displayName is the display name for the output in lieu of the output path in the simulation PKML simulation.

pkParameterList The list of PK parameters for this instance of the SAOutput.

dimension The dimension of this instance of the SAOutput.

unit The display unit of this instance of the SAOutput.

Methods

Public methods:

- SAOutput$new()
- SAOutput$addPKParameter()
- SAOutput$clone()

Method `new()`**:** Create a new `SAOutput` object.

Usage:

```
SAOutput$new(simulation, path, displayName = NULL, unit = NULL)
```

Arguments:

`simulation` simulation A PKML simulation in which the parameter exists.

`path` A PKML simulation in which the output exists.

`displayName` A shorthand string for the output that substitutes for the path for display purposes.

`unit` A valid OSP unit that will be the display unit for this instance of the `SAOutput` object.

Returns: An instance of the `SAOutput` class.

Method `addPKParameter()`**:** Add PK parameter to be evaluated for the `SAOutput` instance.

Usage:

```
SAOutput$addPKParameter(
  standardPKParameter,
  pkParameterDisplayName = NULL,
  startTime = NULL,
  endTime = NULL
)
```

Arguments:

`standardPKParameter` A standard PK parameter selected from among the list of strings in `ospsuite::allPKParameterNames()`.

`pkParameterDisplayName` A shorthand string for the parameter that substitutes for the path for display purposes.

`startTime` Start time within the simulation run for calculation of the PK parameter.

`endTime` End time within the simulation run for calculation of the PK parameter.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
SAOutput$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

SAParameter *SAParameter*

Description

R6 class defining a parameter object

Active bindings

`path` is the path to the parameter within the `simulation` PKML simulation.

`displayName` is the display name for the parameter in lieu of the parameter path in the `simulation` PKML simulation.

`dimension` is the dimension of the parameter in the `simulation` PKML simulation.

`unit` is the unit of the parameter in the `simulation` PKML simulation.

`distribution` is the distribution of the parameter

Methods

Public methods:

- `SAParameter$new()`
- `SAParameter$clone()`

Method `new()`**:** Create a new `SAParameter` object.

Usage:

```
SAParameter$new(
  simulation,
  path,
  displayName = NULL,
  unit = NULL,
  parameterDistribution = NULL,
  defaultVariationRangeForLogUniformDistributions = 0.1
)
```

Arguments:

`simulation` simulation A PKML simulation in which the parameter exists.

`path` A PKML simulation in which the parameter exists.

`displayName` A shorthand string for the parameter that substitutes for the path for display purposes.

`unit` A valid OSP unit used to interpret the numerical values that are input into the `parameterDistribution` object, such as the mean and standard deviation of a normal distribution.

`parameterDistribution` A `SADistribution` object specifying the probability distribution of the parameter.

`defaultVariationRangeForLogUniformDistributions` When no `parameterDistribution` is specified, a loguniform distribution is assumed with multiplicative variation range given by `defaultVariationRangeForLogUniformDistributions`.

Returns: An instance of the `SAParameter` class.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
SAParameter$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`UniformDistribution` *UniformDistribution*

Description

R6 class, child class of `SADistribution`, defining a statistical uniform distribution object

Super class

`ospsuite.globalsensitivity::SADistribution` -> `UniformDistribution`

Active bindings

`type` is the type of the probability distribution.

`minimum` The minimum of the uniform distribution.

`maximum` The maximum of the uniform distribution.

Methods

Public methods:

- `UniformDistribution$new()`
- `UniformDistribution$quantilesToSample()`
- `UniformDistribution$clone()`

Method `new()`**:** Create a new `UniformDistribution` object.

Usage:

```
UniformDistribution$new(minimum, maximum)
```

Arguments:

`minimum` The minimum of the uniform distribution.

`maximum` The maximum of the uniform distribution.

Returns: An instance of the `UniformDistribution` class.

Method `quantilesToSample()`**:** Maps a vector of `quantiles` to corresponding values from the uniform distribution.

Usage:

```
UniformDistribution$quantilesToSample(quantiles)
```

Arguments:

`quantiles` Vector of quantiles.

Returns: A vector of values from the uniform distribution corresponding to the input `quantiles`.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
UniformDistribution$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Index