

Package ‘softwareRisk’

January 21, 2026

Type Package

Title Computation of Node and Path-Level Risk Scores in Scientific Models

Version 0.1.0

Date 2025-12-24

Maintainer Arnald Puy <arnald.puy@pm.me>

Description It leverages the network-like architecture of scientific models together with software quality metrics to identify chains of function calls that are more prone to generating and propagating errors. It operates on `tbl_graph` objects representing call dependencies between functions (callers and callees) and computes risk scores for individual functions and for paths (sequences of function calls) based on cyclomatic complexity, in-degree and betweenness centrality. The package supports variance-based uncertainty and sensitivity analyses after Puy et al. (2022) <[doi:10.18637/jss.v102.i05](https://doi.org/10.18637/jss.v102.i05)> to assess how risk scores change under alternative risk definitions.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Imports `dplyr`, `ggplot2`, `ggraph`, `igraph`, `ineq`, `purrr`, `scales`, `tibble`, `tidygraph`, `grid`, `stats`, `utils`, `rlang`, `data.table`, `sensobol`

Suggests `knitr`, `rmarkdown`, `spelling`

VignetteBuilder `knitr`

Depends `R` (>= 4.1.0)

Language en-US

NeedsCompilation no

Author Arnald Puy [aut, cre] (ORCID: <<https://orcid.org/0000-0001-9469-2156>>)

Repository CRAN

Date/Publication 2026-01-21 19:50:02 UTC

Contents

all_paths_fun	2
gini_index_fun	5
path_fix_heatmap	5
path_uncertainty_plot	7
plot_top_paths_fun	8
risk_form_variance_share_fun	9
slope_fun	11
synthetic_graph	11
theme_AP	12
uncertainty_fun	13

Index

16

all_paths_fun	<i>Enumerate entry-to-sink call paths and compute risk metrics at the node and path level</i>
---------------	---

Description

Given a directed call graph (`tidygraph::tbl_graph`) with a node attribute for cyclomatic complexity, this function:

- computes node-level metrics (in-degree, out-degree, betweenness),
- calculates a node risk score as a weighted combination of rescaled metrics,
- enumerates all simple paths from entry nodes (in-degree = 0) to sink nodes (out-degree = 0),
- computes path-level summaries and a path-level risk score.
- calculates a gini index and the slope of risk at the path-level.

Usage

```
all_paths_fun(
  graph,
  alpha = 0.6,
  beta = 0.3,
  gamma = 0.1,
  risk_form = c("additive", "power_mean"),
  p = 1,
  eps = 1e-12,
  complexity_col = "cyclo",
  weight_tol = 1e-08
)
```

Arguments

graph	A directed <code>tidygraph::tbl_graph</code> . Graph nodes must have a <code>name</code> attribute (i.e., <code>igraph::V(as.igraph(graph))\$name</code>) and a numeric node attribute specified by <code>complexity_col</code> .
alpha, beta, gamma	Numeric non-negative weights for the risk score, constrained such that <code>alpha + beta + gamma == 1</code> (within <code>weight_tol</code>).
risk_form	Character. Form used to combine normalized node metrics into a node risk score. One of "additive" or "power_mean". Default "additive".
p	Numeric scalar. Power parameter used when <code>risk_form = "power_mean"</code> . Must be finite and lie in the interval $[-1, 2]$. Default 1.
eps	Numeric. Small positive constant ϵ used for numerical stability in the $p \rightarrow 0$ (geometric mean) case. Default <code>1e-12</code> .
complexity_col	Character scalar. Name of the node attribute containing cyclomatic complexity. Default "cyclo".
weight_tol	Numeric tolerance for enforcing the weight-sum constraint. Default <code>1e-8</code> .

Details

The normalized node metrics are computed using `scales::rescale()` and denoted by a tilde $\tilde{\cdot}$.

If `risk_form = "additive"`, the risk score for node v_i is computed as

$$r_{(v_i)} = \alpha \tilde{C}_{(v_i)} + \beta \tilde{d}_{(v_i)}^{\text{in}} + \gamma \tilde{b}_{(v_i)},$$

where the weights α , β and γ reflect the relative importance of complexity, coupling and network position, with the constraint $\alpha + \beta + \gamma = 1$.

If `risk_form = "power_mean"`, the risk score for node v_i is computed as the weighted power mean of normalized metrics:

$$r_{(v_i)} = \left(\alpha \tilde{C}_{(v_i)}^p + \beta \tilde{d}_{(v_i)}^{\text{in}\,p} + \gamma \tilde{b}_{(v_i)}^p \right)^{1/p},$$

where p is the power-mean parameter. In the limit $p \rightarrow 0$, this reduces to a weighted geometric mean, implemented with a small constant ϵ to ensure numerical stability:

$$r_{(v_i)} = \exp \left(\alpha \log(\max(\tilde{C}_{(v_i)}, \epsilon)) + \beta \log(\max(\tilde{d}_{(v_i)}^{\text{in}}, \epsilon)) + \gamma \log(\max(\tilde{b}_{(v_i)}, \epsilon)) \right).$$

The path-level risk score is calculated as

$$P_k = 1 - \prod_{i=1}^{n_k} (1 - r_{k(v_i)}),$$

where $r_{k(v_i)}$ is the risk of the i -th function in path k and n_k is the number of functions in that path. The equation behaves like a saturating OR-operator: P_k is at least as large as the maximum individual function risk and monotonically increases as more functions on the path become risky, approaching 1 when several functions have high risk scores.

The Gini index of path k is computed as

$$G_k = \frac{\sum_i \sum_j |r_{k(v_i)} - r_{k(v_j)}|}{2n_k^2 \bar{r}_k},$$

where \bar{r}_k is the mean node risk in path k .

Finally, the trend of risk is defined by the slope of the regression

$$r_{k(v_i)} = \theta_{0k} + \theta_{1k} i + \epsilon_i,$$

where $r_{k(v_i)}$ is the risk score of the function at position i along path k (ordered from upstream to downstream execution) and ϵ_i is a residual term.

The returned `paths` tibble includes `path_cc`, a list-column giving the numeric vector of cyclomatic complexity values in the same order as `path_nodes` / `path_str`.

Value

A named list with two tibbles:

nodes Node-level metrics with columns `name`, `cyclomatic_complexity`, `indeg` (in-degree), `outdeg` (out-degree), `btw` (betweenness), `risk_score`.

paths Path-level metrics with columns `path_id`, `path_nodes`, `path_str`, `hops`, `path_risk_score`, `path_cc`, `gini_node_risk`, `risk_slope`, `risk_mean`, `risk_sum`

Examples

```
# synthetic_graph is a tidygraph::tbl_graph with node attribute "cyclo"
data(synthetic_graph)

# additive risk (default)
out1 <- all_paths_fun(
  graph = synthetic_graph,
  alpha = 0.6, beta = 0.3, gamma = 0.1,
  risk_form = "additive",
  complexity_col = "cyclo"
)

# power-mean risk (p = 0 ~ weighted geometric mean)
out2 <- all_paths_fun(
  graph = synthetic_graph,
  alpha = 0.6, beta = 0.3, gamma = 0.1,
  risk_form = "power_mean",
  p = 0,
  eps = 1e-12,
  complexity_col = "cyclo"
)

out1$nodes
out1$paths
```

gini_index_fun	<i>Compute the Gini index of a numeric vector</i>
----------------	---

Description

Computes the Gini index (a measure of inequality) for a numeric vector. Non-finite (NA, NaN, Inf) values are removed prior to computation. If fewer than two finite values remain, the function returns 0.

Usage

```
gini_index_fun(x)
```

Arguments

x Numeric vector.

Details

The Gini index ranges from 0 (perfect equality) to 1 (maximal inequality).

Value

A numeric scalar giving the Gini index of x.

Examples

```
gini_index_fun(c(1, 1, 1, 1))
gini_index_fun(c(1, 2, 3, 4))
gini_index_fun(c(NA, 1, 2, Inf, 3))
```

path_fix_heatmap	<i>Path-level improvement from fixing high-risk nodes</i>
------------------	---

Description

Compute how much the risk score of the riskiest paths would decrease if selected high-risk nodes were made perfectly reliable (risk fixed to 0), and visualise the result as a heatmap.

Usage

```
path_fix_heatmap(all_paths_out, n_nodes = 20, k_paths = 20)
```

Arguments

all_paths_out	A list returned by <code>all_paths_fun()</code> , with elements <code>nodes</code> and <code>paths</code> . <code>nodes</code> must contain at least the columns <code>name</code> and <code>risk_score</code> . <code>paths</code> must contain at least the columns <code>path_id</code> , <code>path_nodes</code> (list-column of node names) and <code>path_risk_score</code> .
n_nodes	Integer, number of top-risk nodes (by <code>risk_score</code>) to include as rows in the heatmap. Defaults to 20.
k_paths	Integer, number of top-risk paths (by <code>path_risk_score</code>) to include as columns in the heatmap. Defaults to 20.

Details

For each of the top `n_nodes` nodes ranked by `risk_score` and the top `k_paths` paths ranked by `path_risk_score`, the function sets the risk of that node to 0 along the path (for all its occurrences) and recomputes the path risk score under the independence assumption, using

$$P_k = 1 - \prod_{i=1}^{n_k} (1 - r_{k(v_i)})$$

The improvement

$$\Delta P_k = R_{\text{orig}} - R_{\text{fix}}$$

is used as the fill value in the heatmap cells.

Bright cells indicate nodes that act as chokepoints for a given path. Rows with many bright cells correspond to nodes whose refactoring would improve many risky paths (global chokepoints), while columns with a few very bright cells correspond to paths dominated by a single risky node.

Value

A list with two elements:

- `delta_tbl`: a tibble with columns `node`, `path_id` and `deltaR`, containing the reduction in path risk score when fixing the node in that path.
- `plot`: a `ggplot2` object containing the heatmap.

Examples

```
data(synthetic_graph)
out <- all_paths_fun(graph = synthetic_graph, alpha = 0.6, beta = 0.3,
gamma = 0.1, complexity_col = "cyclo")
res <- path_fix_heatmap(all_paths_out = out, n_nodes = 20, k_paths = 20)
res
```

path_uncertainty_plot *Plot path-level uncertainty for the top-risk paths*

Description

Plot the top n_paths paths ranked by their mean risk score, with horizontal error bars representing the uncertainty range (minimum and maximum risk) computed from the Monte Carlo samples stored in uncertainty_analysis.

Usage

```
path_uncertainty_plot(ua_sa_out, n_paths = 20)
```

Arguments

ua_sa_out	A list returned by uncertainty_fun() containing at least an element \$paths, which must be a data frame with columns path_id and uncertainty_analysis. The column uncertainty_analysis is expected to be a list-column where each element is a numeric vector of path risk values obtained from Monte Carlo sampling.
n_paths	Integer, number of top paths (by mean risk) to include in the plot. Defaults to 20.

Details

This function is designed to work with the paths component of the output of [uncertainty_fun\(\)](#). For each path, it summarises the vector of path risk values by computing the mean, minimum and maximum values, and then displays these summaries for the n_paths most risky paths.

Value

A **ggplot2** object.

Examples

```
data(synthetic_graph)
out <- all_paths_fun(graph = synthetic_graph, alpha = 0.6, beta = 0.3,
                      gamma = 0.1, complexity_col = "cyclo")
results <- uncertainty_fun(all_paths_out = out, N = 2^10, order = "first")
path_uncertainty_plot(ua_sa_out = results, n_paths = 20)
```

plot_top_paths_fun *Plot the top risky call paths on a Sugiyama layout*

Description

Visualizes the most risky entry-to-sink paths (by decreasing `path_risk_score`) computed by `all_paths_fun()`. Edges that occur on the top paths are highlighted, with edge colour mapped to the mean path risk and edge width mapped to the number of top paths using that edge. Nodes on the top paths are emphasized, with node size mapped to in-degree and node fill mapped to binned cyclomatic complexity.

Usage

```
plot_top_paths_fun(
  graph,
  all_paths_out,
  model.name = "",
  language = "",
  top_n = 10,
  alpha_non_top = 0.05
)
```

Arguments

<code>graph</code>	A directed <code>tidygraph::tbl_graph</code> representing the call graph to plot (typically the same graph used as input to <code>all_paths_fun()</code>).
<code>all_paths_out</code>	Output from <code>all_paths_fun()</code> , i.e. a list with elements <code>nodes</code> and <code>paths</code> . For backward compatibility, a <code>paths</code> tibble can also be supplied directly; in that case node metrics are derived from <code>graph</code> where possible.
<code>model.name</code>	Character scalar used in the plot title (e.g., model name).
<code>language</code>	Character scalar used in the plot title (e.g., language name).
<code>top_n</code>	Integer. Number of highest-risk paths to display (default 10).
<code>alpha_non_top</code>	Numeric in [0, 1] . Alpha (transparency) for edges that are not on the top-risk paths. Smaller values fade background edges more.

Details

The function selects the `top_n` paths by sorting `paths_tbl` on `path_risk_score` (descending). For those paths, it:

- builds an edge list from `path_nodes`,
- marks graph edges that appear on at least one top path,
- computes `path_freq` (how many top paths include each edge),

- computes `risk_mean_path` (mean of `risk_sum` across top paths that include each edge),
- highlights nodes that appear on any top path.

Node fills are based on `cyclomatic_complexity` using breaks $(-\text{Inf}, 10], (10, 20], (20, 50], (50, \text{Inf}]$ as per Watson & McCabe (1996).

This function relies on external theming/label objects `theme_AP()` and `lab_expr` being available in the calling environment or package namespace.

Value

A ggplot object (invisibly). The plot is also printed as a side effect.

References

Watson, A. H. and McCabe, T. J. (1996). *Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric*. NIST Special Publication 500-235, National Institute of Standards and Technology, Gaithersburg, MD. doi:10.6028/NIST.SP.500-235

Examples

```
data(synthetic_graph)
out <- all_paths_fun(graph = synthetic_graph, alpha = 0.6, beta = 0.3,
gamma = 0.1, complexity_col = "cyclo")
p <- plot_top_paths_fun(synthetic_graph, out, model.name = "MyModel", language = "R", top_n = 10)
p
```

risk_form_variance_share_fun

Proportion of risk-score variance due to functional-form uncertainty

Description

Computes, for each node, the proportion of variance in the node risk score that is attributable to uncertainty in the *functional form* used to construct the risk score (additive vs power-mean), as opposed to uncertainty in the parameters within a given form.

Usage

```
risk_form_variance_share_fun(all_paths_out, N, q = 0.5, eps = 1e-12)
```

Arguments

<code>all_paths_out</code>	A list produced by <code>all_paths_fun()</code> with element <code>nodes</code> . <code>nodes</code> must contain columns <code>name</code> , <code>cyclomatic_complexity</code> , <code>indeg</code> , and <code>btw</code> .
<code>N</code>	Integer. Base sample size passed to <code>sensobol::sobol_matrices()</code> .
<code>q</code>	Numeric scalar in $[0, 1]$. Mixture weight giving the probability that the functional form is "power_mean" (and $1 - q$ for "additive"). Default 0.5.

eps	Numeric. Small positive constant ϵ used for numerical stability in the $p \rightarrow 0$ evaluation. Default 1e-12.
-----	--

Details

The function draws parameter samples for two model forms:

- Additive form with weights (α, β, γ) normalized to sum to one.
- Power-mean form with weights (α, β, γ) normalized to sum to one and a power parameter p mapped to the interval $[-1, 2]$.

For each node, it evaluates the risk under both forms and applies the variance decomposition for a two-component mixture model:

$$\text{Var}(Y) = \mathbb{E} [\text{Var}(Y \mid M)] + \text{Var} (\mathbb{E}[Y \mid M]) ,$$

where M indicates the functional form. Let q be the probability of choosing the power-mean form (and $1 - q$ the probability of choosing the additive form). The between-form component is

$$V_{\text{between}} = q(1 - q) (\mu_1 - \mu_0)^2 ,$$

and the total variance is

$$V_{\text{total}} = (1 - q)V_0 + qV_1 + V_{\text{between}} ,$$

where μ_0, V_0 are the mean and variance under the additive form and μ_1, V_1 are the mean and variance under the power-mean form. The reported proportion is $V_{\text{between}}/V_{\text{total}}$.

This function focuses only on variance attributable to the *choice of functional form*. It does not return full uncertainty draws, Sobol sensitivity indices, or path-level uncertainty propagation.

Value

A `data.table` with columns:

name Node name.

form_variance_share Proportion of node-risk variance attributable to functional-form uncertainty.

Examples

```
data(synthetic_graph)
out <- all_paths_fun(graph = synthetic_graph, complexity_col = "cyclo")
vf <- risk_form_variance_share_fun(all_paths_out = out, N = 2^10, q = 0.5)
vf
```

slope_fun*Compute the linear slope of a numeric sequence*

Description

Computes the slope of a simple linear regression of a numeric vector against its index (seq_along(x)). Non-finite (NA, NaN, Inf) values are removed prior to computation. If fewer than two finite values remain, the function returns 0.

Usage

```
slope_fun(x)
```

Arguments

x	Numeric vector.
---	-----------------

Details

The slope is estimated from the model $x_i = \beta_0 + \beta_1 i + \varepsilon_i$, where $i = 1, \dots, n$. The function returns the estimated slope β_1 .

This summary is useful for characterizing monotonic trends in ordered risk values along a path.

Value

A numeric scalar giving the slope of the fitted linear trend.

Examples

```
slope_fun(c(1, 2, 3, 4))
slope_fun(c(4, 3, 2, 1))
slope_fun(c(NA, 1, 2, Inf, 3))
```

synthetic_graph*Synthetic citation graph for software risk examples*

Description

A synthetic directed graph with cyclomatic complexity values to illustrate the functions of the package.

Usage

```
data(synthetic_graph)
```

Format

A `tbl_graph` with:

nodes 55 nodes
edges 122 directed edges

Details

The graph is stored as a `tbl_graph` object with:

- Node attributes: `name`, `cyclo`
- Directed edges defined by `from` → `to`

theme_AP

A clean, publication-oriented ggplot2 theme

Description

`theme_AP()` provides a minimalist, publication-ready theme based on `ggplot2::theme_bw()`, with grid lines removed, compact legends, and harmonized text sizes. It is designed for dense network and path-visualization plots (e.g. call graphs, risk paths).

Usage

`theme_AP()`

Details

The theme:

- removes major and minor grid lines,
- uses transparent legend backgrounds and keys,
- standardizes text sizes for axes, legends, strips, and titles,
- reduces legend spacing for compact layouts.

This theme is intended to be composable: it should be added to a `ggplot` object using `+ theme_AP()`.

Value

A `ggplot2::theme` object.

Examples

```
ggplot2::ggplot(mtcars, ggplot2::aes(mpg, wt)) +
  ggplot2::geom_point() +
  theme_AP()
```

Description

Runs a full variance-based uncertainty and sensitivity analysis for node risk scores using the results returned by `all_paths_fun()` and the functions provided by the `sensobol` package (Puy et al. 2022).

Usage

```
uncertainty_fun(
  all_paths_out,
  N,
  order,
  risk_form = c("additive", "power_mean")
)
```

Arguments

<code>all_paths_out</code>	A list produced by <code>all_paths_fun()</code> with elements <code>nodes</code> and <code>paths</code> . <code>nodes</code> must contain columns <code>name</code> , <code>cyclomatic_complexity</code> , <code>indeg</code> , <code>btw</code> ; <code>paths</code> must contain <code>path_id</code> , <code>path_nodes</code> , <code>path_str</code> , and <code>hops</code> .
<code>N</code>	Integer. Base sample size used for Sobol' matrices.
<code>order</code>	Passed to <code>sensobol::sobol_matrices()</code> and <code>sensobol::sobol_indices()</code> to control which Sobol indices are computed (e.g., first/total/second order), depending on your implementation.
<code>risk_form</code>	Character. Risk definition used in the uncertainty/sensitivity analysis. One of "additive" or "power_mean". Default "additive".

Details

To assess how sensitive the risk scores are to the choice of weighting parameters, this function explores many alternative combinations of weights and (optionally) a power-mean parameter, and examines how resulting risk scores vary.

When `risk_form = "additive"`, uncertainty is induced by sampling weight triplets (α, β, γ) under the constraint $\alpha + \beta + \gamma = 1$, representing different plausible balances between complexity, connectivity and centrality.

When `risk_form = "power_mean"`, uncertainty is induced by sampling both the weights (α, β, γ) (renormalized to sum to 1) and a power parameter p used in the node-risk definition:

$$r = \left(\alpha \tilde{C}^p + \beta (\tilde{d}^{\text{in}})^p + \gamma \tilde{b}^p \right)^{1/p}.$$

For each node, risk scores are repeatedly recalculated using the sampled parameter combinations, producing a distribution of possible outcomes. This distribution is then used to quantify uncertainty in the risk scores and compute Sobol' sensitivity indices for each sampled parameter.

Path-level uncertainty is obtained by propagating node-level uncertainty draws through the path aggregation function:

$$P_k = 1 - \prod_{i=1}^{n_k} (1 - r_{k(v_i)}) ,$$

where $r_{k(v_i)}$ are node risks along path k .

All uncertainty metrics are computed from the first N Sobol draws (matrix A), while sensitivity indices use the full Sobol' design.

For more information about the uncertainty and sensitivity analysis and the output of this function, see the **sensobol** package (Puy et al. 2022).

The returned node table includes the following columns:

- **name**: name of the node.
- **uncertainty_analysis**: numeric vector giving the uncertainty draws in the node risk score.
- **sensitivity_analysis**: object returned by `sensobol::sobol_indices()` (per-node).

The returned paths table includes:

- **path_id**: path identifier.
- **path_str**: sequence of function calls for each path.
- **hops**: number of edges.
- **uncertainty_analysis**: numeric vector giving the uncertainty draws in the path risk score.
- **gini_index**: numeric vector giving the uncertainty draws in the gini index.
- **risk_trend**: numeric vector giving the uncertainty draws in the risk trend.

Value

A named list with:

nodes A data.table of node results.

paths A data.table of path results.

References

Puy, A., Lo Piano, S., Saltelli, A., and Levin, S. A. (2022). *sensobol: An R Package to Compute Variance-Based Sensitivity Indices*. Journal of Statistical Software, 102(5), 1–37. doi:10.18637/jss.v102.i05

Examples

```
data(synthetic_graph)
out <- all_paths_fun(graph = synthetic_graph, alpha = 0.6, beta = 0.3,
                      gamma = 0.1, complexity_col = "cyclo")

# Additive risk (increase N to at least 2^10 for a proper UA/SA)
results1 <- uncertainty_fun(all_paths_out = out, N = 2^2, order = "first",
                             risk_form = "additive")

# Power-mean risk (increase N to at least 2^10 for a proper UA/SA)
```

```
results2 <- uncertainty_fun(all_paths_out = out, N = 2^2, order = "first",
                           risk_form = "power_mean")  
  
results1$nodes  
results1$paths
```

Index

* **datasets**
 synthetic_graph, 11

 all_paths_fun, 2
 all_paths_fun(), 6, 8, 9, 13

 ggplot2::theme_bw(), 12
 gini_index_fun, 5

 path_fix_heatmap, 5
 path_uncertainty_plot, 7
 plot_top_paths_fun, 8

 risk_form_variance_share_fun, 9

 slope_fun, 11
 synthetic_graph, 11

 theme_AP, 12

 uncertainty_fun, 13
 uncertainty_fun(), 7