Package 'softwareRisk'

April 4, 2026

Type Package

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

Version 0.2.0

Date 2026-04-03

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> 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, sensobol

Suggests knitr, rmarkdown, spelling, testthat (>= 3.0.0)

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 2026-04-04 20:20:02 UTC

Contents

`all_paths_fun` *Enumerate entry-to-sink call paths and compute risk metrics at the node and path level*

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,
  p = 1,
  eps = 1e-12,
  complexity_col = "cyclo",
  weight_tol = 1e-08
)
```

Arguments

graph	A directed tidygraph::tbl_graph. Graph nodes must have a name attribute (i.e., igraph::V(as.igraph(graph))$name) and a numeric node attribute specified by complexity_col.
alpha, beta, gamma	Numeric non-negative weights for the risk score, constrained such that alpha + beta + gamma == 1 (within weight_tol).
p	Numeric scalar. Power parameter for the weighted power mean. Must be finite and lie in the interval $[-1, 2]$. When $p = 1$ the formula reduces to a weighted sum. Default 1.
eps	Numeric. Small positive constant ϵ used for numerical stability in the $p \to 0$ (geometric mean) case. Default 1e-12.
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 1e-8.

Details

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

The risk score for node v_i is computed as the weighted power mean of normalized metrics:

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

where p is the power-mean parameter. When $p = 1$ this reduces to a weighted sum (additive). In the limit $p \to 0$, this reduces to a weighted geometric mean, implemented with a small constant ϵ to ensure numerical stability:

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

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

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

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 where each element is the vector of per-node cyclomatic complexity values along the path.

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 (p = 1, default)
out1 <- all_paths_fun(
  graph = synthetic_graph,
  alpha = 0.6, beta = 0.3, gamma = 0.1,
  p = 1,
  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,
  p = 0,
  eps = 1e-12,
  complexity_col = "cyclo"
)

out1$nodes
out1$paths
```

gini_index_fun *Compute the Gini index of a numeric vector*

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 *Path-level improvement from fixing high-risk nodes*

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

graph	A directed tidygraph::tbl_graph representing the call graph to plot (typically the same graph used as input to all_paths_fun()).
all_paths_out	Output from all_paths_fun(), i.e. a list with elements nodes and paths. For backward compatibility, a paths tibble can also be supplied directly; in that case node metrics are derived from graph where possible.
model.name	Character scalar used in the plot title (e.g., model name).
language	Character scalar used in the plot title (e.g., language name).
top_n	Integer. Number of highest-risk paths to display (default 10).
alpha_non_top	Numeric between 0 and 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 `(-Inf, 10]`, `(10, 20]`, `(20, 50]`, `(50, 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
```

---

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 $i = 1, \ldots, n$. 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()
```

uncertainty_fun *Uncertainty and sensitivity analysis of node and path risk*

Description

Runs a full variance-based uncertainty and sensitivity analysis (UA/SA) 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, eps = 1e-12)
```

Arguments

`all_paths_out`	A list produced by `all_paths_fun()` with elements `nodes` and `paths`. `nodes` must contain columns `name`, `cyclomatic_complexity`, `indeg`, `btw`; `paths` must contain `path_id`, `path_nodes`, `path_str`, and `hops`.
`N`	Integer. Base sample size used for Sobol' matrices.
`order`	Passed to `sensobol::sobol_matrices()` and `sensobol::sobol_indices()` to control which Sobol indices are computed (e.g., first/total/second order), depending on your implementation.
`eps`	Numeric. Small positive constant ϵ used for numerical stability in the $p \to 0$ evaluation. Default `1e-12`.

Details

Uncertainty is induced by jointly sampling the weights (α, β, γ) (renormalized to sum to 1) and the power parameter $p \in [-1, 2]$ 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. Sobol' first-order and/or total-order sensitivity indices are then computed for all four parameters (α, β, γ, and p), quantifying how much of the variance in the node risk score is attributable to each parameter.

Parameter labels and the Sobol' design. Internally the design samples four independent $U(0, 1)$ values (`a_raw`, `b_raw`, `c_raw`, `p_raw`) because the Sobol' quasi-random sequence requires independent uniform inputs. Before evaluating the risk model, the raw draws are transformed: the three weight draws are normalised to sum to one, yielding α, β, γ; and `p_raw` is mapped linearly to $p \in [-1, 2]$. The sensitivity indices are then attributed to the *transformed* parameters and the output labels them as `alpha`, `beta`, `gamma`, and `p` rather than the internal raw names, so the results are directly interpretable in terms of the model parameters.

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 of length N giving the uncertainty draws in the node risk score (from Sobol matrix A).
- `sensitivity_analysis`: object returned by `sensobol::sobol_indices()` for that node, containing Sobol' indices labelled `alpha`, `beta`, `gamma`, and `p`. These correspond to the normalised weights and the power-mean exponent respectively. The indices are computed on the transformed parameters (see Details).

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 tibble of node results.

paths A tibble 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")

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

results$nodes
results$paths
```

Index