Package 'evaluator'

October 18, 2018

Title Quantified Risk Assessment Toolkit

Version 0.3.1

Description An open source risk analysis toolkit based on the OpenFAIR taxonomy <https://www2.opengroup.org/ogsys/catalog/C13K> and risk assessment standard <https://www2.opengroup.org/ogsys/catalog/C13G>. Empowers an organization to perform a quantifiable, repeatable, and data-driven risk review.

Depends R (>= 3.3.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports dplyr, extrafont, ggplot2, mc2d, furrr, purrr, readr, readxl, rlang, scales, stringi, tibble, tidyr, viridis

RoxygenNote 6.1.0

Suggests DT, pander (>= 0.6.1), psych, ggalt (>= 0.4.0), flexdashboard (>= 0.4), forcats, statip, knitr, purrrlyr, rmarkdown (>= 1.9), shiny, testthat, shinytest, EnvStats, covr, mockery

SystemRequirements pandoc

VignetteBuilder knitr

URL https://evaluator.severski.net

BugReports https://github.com/davidski/evaluator/issues

NeedsCompilation no

Author David Severski [aut, cre] (<https://orcid.org/0000-0001-7867-0459>)

Maintainer David Severski <davidski@deadheaven.com>

Repository CRAN

Date/Publication 2018-10-18 15:40:03 UTC

R topics documented:

calculate_max_losses *Calculate maximum losses*

Description

Calculate the biggest single annual loss for each scenario, as well as the minimum and maximum ALE across all simulations. Calculations both with and without outliers (if passed) are returned.

Usage

```
calculate_max_losses(simulation_results, scenario_outliers = NULL)
```

Arguments

simulation_results	Simulation results dataframe.
scenario_outliers	Optional vector of IDs of outlier scenarios.

Value

A dataframe with the following columns:

- `scenario_id` - index of the simulation
- `biggest_single_scenario_loss` - the biggest annual loss in that simulation,
- `min_loss` - the smallest annual loss in that simulation,
- `max_loss` - the total annual losses in that simulation
- `outliers` - logical of whether or not outliers are included

Examples

```
data(simulation_results)
calculate_max_losses(simulation_results)
```

capabilities *Capabilities*

Description

A qualitative dataset of sample capabilities and qualitative level of effectiveness.

Usage

```
capabilities
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 60 rows and 4 columns.

Details

capability_id unique id of the capability

domain_id domain id to which the capability applies

capability full text summary of the capability

diff qualitative label of control effectiveness

`compare_tef_vuln` *Calculate number of loss events which occur in a simulated period*

Description

Composition function for use in `sample_lef`. Given a count of the number of threat events (TEF) and the level of vulnerability (as a percentage), calculate how many of those become loss events (LEF).

Usage

```
compare_tef_vuln(tef, vuln)
```

Arguments

`tef`	Threat event frequency (n).
`vuln`	Vulnerability (percentage).

Value

List containing samples (as a vector) and details (as a list).

See Also

Other OpenFAIR helpers: `get_mean_control_strength`, `openfair_tef_tc_diff_lm`, `sample_diff`, `sample_lef`, `sample_lm`, `sample_tc`, `sample_tef`, `sample_vuln`, `select_loss_opportunities`

Examples

```
compare_tef_vuln(tef = 500, vuln = .25)
```

create_templates *Create initial template files*

Description

Given a base directory, copy the provided sample files into an `inputs` subdirectory. This makes the starter files available for customizing and data collection. The `inputs` directory will be created if not already present. Preexisting files, if present, will not be overwritten. Also creates an empty `results` subdirectory as a default location for evaluator output.

Usage

```
create_templates(base_directory = "~/evaluator")
```

Arguments

`base_directory`	Parent directory under which to create starter files.

Value

A dataframe of the starter filenames, along with a flag on whether a file was copied.

Examples

```
## Not run:
create_templates("~/evaluator")

## End(Not run)
```

derive_controls *Derive control difficulty parameters for a given qualitative scenario*

Description

Given a comma-separated list of control IDs in a scenario, identify the qualitative rankings associated with each scenario, convert to their quantitative parameters, and return a dataframe of the set of parameters.

Usage

```
derive_controls(capability_ids, capabilities, mappings)
```

Arguments

`capability_ids`	Comma-delimited list of capabilities in scope for a scenario.
`capabilities`	Dataframe of master list of all qualitative capabilities.
`mappings`	Qualitative mappings dataframe.

Value

A dataframe of quantitative estimate parameters for the capabilities applicable to a given scenario.

Examples

```
data(capabilities)
capability_ids <- c("1, 3")
mappings <- data.frame(type = "diff", label = "1 - Immature", l = 0, ml = 2, h = 10,
                       conf = 3, stringsAsFactors = FALSE)
derive_controls(capability_ids, capabilities, mappings)
```

---

dollar_millions *Format dollar amounts in terms of millions of USD*

Description

Given a number, return a string formatted in terms of millions of dollars.

Usage

```
dollar_millions(x)
```

Arguments

x	A number.

Value

String in the format of $xM.

Examples

```
dollar_millions(1.523 * 10^6)
```

---

domains *Domain mappings*

Description

A dataset of domains and domain IDs.

Usage

```
domains
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 14 rows and 2 columns.

Details

domain_id abbreviated name of the domain

domain full title of the domain

`domain_summary` *Domain-level risk summary*

Description

A dataset of quantified information security risk, summarized at the domain level.

Usage

```
domain_summary
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 14 rows and 13 columns.

Details

domain_id abbreviated name of the domain

domain full title of the domain

ale annual loss expected, in US dollars

simulation simulation id number

`encode_scenarios` *Encode qualitative data to quantitative parameters*

Description

Given an input of:

- qualitative risk scenarios
- qualitative capabilities
- translation table from qualitative labels to quantitative parameters

Usage

```
encode_scenarios(scenarios, capabilities, mappings)
```

Arguments

`scenarios`	Qualitative risk scenarios dataframe.
`capabilities`	Qualitative program capabilities dataframe.
`mappings`	Qualitative to quantitative mapping dataframe.

Details

Create a unified dataframe of quantitative scenarios ready for simulation.

Value

A dataframe of capabilities for the scenario and parameters for quantified simulation.

Examples

```
data(qualitative_scenarios, capabilities, mappings)
encode_scenarios(qualitative_scenarios, capabilities, mappings)
```

`evaluator` `evaluator` *package*

Description

Quantified Information Risk Simulation Toolkit

Details

See the online documentation located at https://evaluator.severski.net/

`explore_scenarios` *Launch the Scenario Explorer web application*

Description

Evaluator provides a simple Shiny-based web application for interactive exploration of simulation results. This allows a user to interactively review simulation output without generating an extensive report. For users comfortable with R, working directly with the result dataframes will usually be preferable, with the Explorer application provided as a bare-bones data exploration tool.

Usage

```
explore_scenarios(input_directory = "~/evaluator/inputs",
  results_directory = "~/evaluator/results", styles = NULL,
  intermediates_dir = tempdir(), quiet = TRUE, ...)
```

Arguments

input_directory	Location of input files.
results_directory	Location of simulation results.
styles	Optional full path to CSS file to override default styles.
intermediates_dir	Location for intermediate knit files.
quiet	TRUE to suppress printing of pandoc output.
...	Any other parameters to pass straight to rmarkdown::run.

Value

Invisible NULL.

Examples

```
## Not run:
explore_scenarios("~/inputs", "~/results")

## End(Not run)
```

---

generate_event_outcomes_plot

Display the distribution of threat events contained vs. realized across all domains

---

Description

Creates a barbell plot showing the number and percentage of events contained (not resulting in loss) vs the number and percentage of loss events (threat events resulting in losses).

Usage

```
generate_event_outcomes_plot(domain_summary)
```

Arguments

domain_summary	Domain-level summary from domain_summary.

Value

ggplot object.

Examples

```
data(domain_summary)
generate_event_outcomes_plot(domain_summary)
```

generate_heatmap *Display a heatmap of impact by domain*

Description

Given a domain_summary and a list of all domains, generate a heatmap colored by the 95 greater than others.

Usage

```
generate_heatmap(domain_summary)
```

Arguments

domain_summary	Simulations summarized at a domain level via summarize_domains.

Value

A ggplot object.

Examples

```
data(domain_summary)
generate_heatmap(domain_summary)
```

generate_report *Generate sample analysis report*

Description

Given a set of input files and summarized simulation results, create a skeleton risk analysis report. This report attempts to summarize the results of the analysis at a top level, using 95 metric, while also providing more detailed analysis at both a per-domain and per-scenario level.

Usage

```
generate_report(input_directory = "~/evaluator/inputs",
  results_directory = "~/evaluator/results", output_file,
  styles = NULL, focus_scenario_ids = c(51, 12), format = "html",
  intermediates_dir = tempdir(), quiet = TRUE, ...)
```

Arguments

`input_directory`	Location of input files.
`results_directory`	Location of simulation results.
`output_file`	Full path to output file.
`styles`	Optional full path to CSS file to override default styles.
`focus_scenario_ids`	IDs of scenarios of special interest.
`format`	Format to generate (html, pdf, word).
`intermediates_dir`	Location for intermediate knit files.
`quiet`	TRUE to suppress printing of pandoc output.
`...`	Any other parameters to pass straight to `rmarkdown::render`.

Details

This report includes several sections where an analyst will need to modify and fill in details for their specific organization. Of particular note is the Recommendations section, which will always need to be updated.

Value

Default return values of the `rmarkdown::render` function.

Examples

```
## Not run:
generate_report("~/inputs", "~/results", "~/risk_report.html")

## End(Not run)
```

`generate_scatterplot` *Display a scatterplot for a particular scenario ID*

Description

Given a detailed results dataframe and a specific scenario identifier, create a scatterplot of the number of loss events versus the annual loss expected. This provides a detailed view on the results for a particular scenario.

Usage

```
generate_scatterplot(simulation_results, scenario_id)
```

Arguments

simulation_results	Simulation results from run_simulations.
scenario_id	ID of the scenario to display.

Value

A ggplot object.

Examples

```
data(simulation_results)
generate_scatterplot(simulation_results, scenario_id = 50)
```

---

get_base_fontfamily *Select a base graphics font family*

Description

The Benton Sans Regular font is preferred with a fallback of Arial Narrow. If neither font is available, use a default sans family font.

Usage

```
get_base_fontfamily()
```

Value

String of the preferred base font.

Examples

```
get_base_fontfamily()
```

---

get_mean_control_strength *Calculate difficulty strength across multiple controls by taking the mean*

Description

Given a set of estimation parameters, calculate control strength as the arithmetic mean of sampled control effectiveness.

Usage

```
get_mean_control_strength(n, diff_parameters)
```

Arguments

n	Number of threat events to sample controls across.
diff_parameters	Parameters to pass to sample_diff.

Value

Vector of control effectiveness.

See Also

Other OpenFAIR helpers: compare_tef_vuln, openfair_tef_tc_diff_lm, sample_diff, sample_lef, sample_lm, sample_tc, sample_tef, sample_vuln, select_loss_opportunities

---

import_capabilities *Import capabilities from survey spreadsheet*

Description

Import capabilities from survey spreadsheet

Usage

```
import_capabilities(survey_file = system.file("survey", "survey.xlsx",
  package = "evaluator"), domains)
```

Arguments

survey_file	Path to survey XLSX file. If not supplied, a default sample file is used.
domains	Dataframe of domains and domain IDs.

Value

Extracted capabilities as a dataframe.

Examples

```
data(domains)
import_capabilities(domains = domains)
```

import_scenarios *Import scenarios from survey spreadsheet*

Description

Import scenarios from survey spreadsheet

Usage

```
import_scenarios(survey_file = system.file("survey", "survey.xlsx",
  package = "evaluator"), domains)
```

Arguments

survey_file	Path to survey XLSX file. Defaults to a sample file if not supplied.
domains	Dataframe of domains and domain IDs.

Value

Extracted qualitative scenarios as a dataframe.

Examples

```
data(domains)
import_scenarios(domains = domains)
```

import_spreadsheet *Import the scenario spreadsheet*

Description

This is a wrapper function around import_scenarios and import_capabilities. When invoked, the output of both functions are written to disk.

Usage

```
import_spreadsheet(survey_file = system.file("survey", "survey.xlsx",
  package = "evaluator"), domains = NULL,
  output_dir = "~/evaluator/results")
```

Arguments

survey_file	Path to survey XLSX file. Defaults to an Evaluator-provided sample spreadsheet.
domains	Dataframe of domains and domain IDs. Defaults to built-in sample domains dataset.
output_dir	Output file directory. Defaults to a data subdirectory in the current working directory.

Value

Dataframe of generated files (capabilities.csv and scenarios.csv)

`load_data` *Load input and results files*

Description

Given a input directory and a directory of simulation results, load all of the key Evaluator data objects into memory.

Usage

```
load_data(input_directory = "~/evaluator/inputs",
  results_directory = "~/evaluator/results")
```

Arguments

`input_directory`
: Location of input files.

`results_directory`
: Location of simulation results.

Value

List of all key data objects.

Examples

```
## Not run:
load_data("~/evaluator/inputs", "~/evaluator/results")

## End(Not run)
```

`mappings` *Qualitative to quantitative mappings*

Description

A dataset of sample mappings from qualitative labels to quantitative distribution parameters.

Usage

```
mappings
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 14 rows and 6 columns.

Details

type OpenFAIR taxonomy to which this mapping applies

label Qualitative label

l BetaPERT low value

ml BetaPERT most likely value

h BetaPERT high value

conf BetaPERT confidence value

`openfair_example` *Launch OpenFAIR demonstration web application*

Description

A simple web application to demonstrate OpenFAIR modeling. This application allows a user to enter beta PERT parameters and run simulations to see the distribution of results, with high level summary statistics. As a demonstration application, only TEF+TC+DIFF+LM parameters may be entered.

Usage

```
openfair_example(intermediates_dir = tempdir(), quiet = TRUE)
```

Arguments

`intermediates_dir`	Location for intermediate knit files.
`quiet`	TRUE to suppress printing of pandoc output.

Value

Invisible NULL

Examples

```
## Not run:
openfair_example()

## End(Not run)
```

`openfair_tef_tc_diff_lm` *Run an OpenFAIR simulation at the TEF/TC/DIFF/LM levels*

Description

Run an OpenFAIR model with parameters provided for TEF, TC, DIFF, and LM sampling. If there are multiple controls provided for a scenarios, the arithmetic mean (average) is taken across samples for all controls to get the effective control strength for a given simulation.

Usage

```
openfair_tef_tc_diff_lm(scenario, n = 10^4, title = "Untitled",
  verbose = FALSE)
```

Arguments

`scenario`	List of TEF, TC, and LM parameters.
`n`	Number of simulations to run.
`title`	Optional name of scenario.
`verbose`	Whether to print progress indicators.

Value

Dataframe of scenario name, threat_event count, loss_event count, mean TC and DIFF exceedance, and ALE samples.

See Also

Other OpenFAIR helpers: `compare_tef_vuln`, `get_mean_control_strength`, `sample_diff`, `sample_lef`, `sample_lm`, `sample_tc`, `sample_tef`, `sample_vuln`, `select_loss_opportunities`

Examples

```
data(quantitative_scenarios)
scenario <- quantitative_scenarios[1, ]
openfair_tef_tc_diff_lm(scenario, 10)
```

qualitative_scenarios *Qualitative information security risk scenarios*

Description

A dataset of qualified information security risk scenarios.

Usage

```
qualitative_scenarios
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 56 rows and 8 columns.

Details

scenario_id id of the scenario, primary key

scenario full text description of the risk scenario

tcomm full text name of threat community

tef qualitative label of the frequency of threat events

tc qualitative label of the threat capability

lm qualitative label of the single loss magnitude

domain_id domain id to which the scenario applies

controls comma separate string of the controls for the scenario

quantitative_scenarios *Quantified information risk scenarios*

Description

A dataset of quantified risk scenarios, with parameters describing the distribution of each input.

Usage

```
quantitative_scenarios
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 56 rows and 9 columns.

Details

scenario full text description of the risk scenario

scenario_id id of the scenario, primary key

tcomm full text name of threat community

domain_id domain abbreviation

controls comma separated list of control ids that apply to this scenario

diff_params nested dataframe of the controls and difficulty parameters associated with the scenario

tef_params list of the threat expected frequency parameters

tc_params list of the threat capability parameters

lm_params list of the loss magnitude parameters

---

read_quantitative_inputs *Load quantitative inputs*

Description

Given an input directory, load the key quantitative objects into memory. The key quantitative inputs for Evaluator processing include: * domains - domains and domain_ids * risk_tolerances - the risk tolerances of the organization * quantitative_scenarios - risk scenarios and quantified parameters

Usage

```
read_quantitative_inputs(input_directory = "~/evaluator/inputs")
```

Arguments

`input_directory` Location of input files.

Value

List of domains, quantitative_scenarios, and risk_tolerances

Examples

```
## Not run:
read_quantitative_inputs("~/evaluator/inputs")

## End(Not run)
```

risk_dashboard *Launch a single page summary risk dashboard*

Description

Given the input files and the analysis summary file, create a basic one- page summary with an overview of the results per domain and scenario. Intended as a skeleton showing how the results could be displayed at an executive level.

Usage

```
risk_dashboard(input_directory = "~/evaluator/inputs",
  results_directory = "~/evaluator/results", output_file,
  intermediates_dir = tempdir(), quiet = TRUE, ...)
```

Arguments

input_directory	Location of input files
results_directory	Location of simulation results
output_file	Full path to the desired output file.
intermediates_dir	Location for intermediate knit files.
quiet	TRUE to suppress printing of pandoc output.
...	Any other parameters to pass straight to rmarkdown::render

Value

Default return values of the rmarkdown::render function.

Examples

```
## Not run:
risk_dashboard("~/inputs", "~/simulations")

## End(Not run)
```

`run_simulations` *Run simulations for all scenarios*

Description

Given a dataframe of quantitative scenarios, run an OpenFAIR Monte Carlo simulation for each scenario, returning a combined dataframe of all results.

Usage

```
run_simulations(scenario, simulation_count = 10000L,
  model = "openfair_tef_tc_diff_lm", ale_maximum = NULL,
  verbose = FALSE)
```

Arguments

`scenario`	Quantitative scenarios.
`simulation_count`	Number of simulations for each scenario.
`model`	OpenFAIR model to use.
`ale_maximum`	Apply a maximum per year, per simulation, loss maximum.
`verbose`	Whether verbose console output is requested.

Value

Dataframe of raw results.

Examples

```
data(quantitative_scenarios)
# run a single scenario in a trivial number (10) of trials
run_simulations(quantitative_scenarios[1, ], 10)
```

`sample_diff` *Calculate the difficulty presented by controls, given a function and parameters for that function*

Description

Calculate the difficulty presented by controls, given a function and parameters for that function

Usage

```
sample_diff(func = NULL, params = NULL)
```

Arguments

func	Function to use to simulate DIFF, defaults to rpert.
params	Optional parameters to pass to func.

Value

List containing type ("diff"), samples (as a vector), and details (as a list).

See Also

Other OpenFAIR helpers: compare_tef_vuln, get_mean_control_strength, openfair_tef_tc_diff_lm, sample_lef, sample_lm, sample_tc, sample_tef, sample_vuln, select_loss_opportunities

---

sample_lef *Sample loss event frequency*

Description

Sample loss event frequency

Usage

```
sample_lef(func = NULL, params = NULL)
```

Arguments

func	Function to use to simulate LEF, defaults to rnorm.
params	Optional parameters to pass to func

Value

List containing type ("lef"), samples (as a vector), and details (as a list).

See Also

Other OpenFAIR helpers: compare_tef_vuln, get_mean_control_strength, openfair_tef_tc_diff_lm, sample_diff, sample_lm, sample_tc, sample_tef, sample_vuln, select_loss_opportunities

sample_lm — *Given a number of loss events and a loss distribution, calculate losses*

Description

Given a number of loss events and a loss distribution, calculate losses

Usage

```
sample_lm(func = NULL, params = NULL)
```

Arguments

func	Function to use to simulate TEF, defaults to rpert.
params	Optional parameters to pass to func.

Value

List containing type ("lm"), samples (as a vector), and details (as a list).

See Also

Other OpenFAIR helpers: compare_tef_vuln, get_mean_control_strength, openfair_tef_tc_diff_lm, sample_diff, sample_lef, sample_tc, sample_tef, sample_vuln, select_loss_opportunities

sample_tc — *Sample threat capabilities (TC) from a distribution function*

Description

Sample threat capabilities (TC) from a distribution function

Usage

```
sample_tc(func = NULL, params = NULL)
```

Arguments

func	Function to use to simulate TC, defaults to rpert.
params	Optional parameters to pass to func.

Value

List containing type ("tc"), samples (as a vector), and details (as a list).

See Also

Other OpenFAIR helpers: compare_tef_vuln, get_mean_control_strength, openfair_tef_tc_diff_lm, sample_diff, sample_lef, sample_lm, sample_tef, sample_vuln, select_loss_opportunities

---

`sample_tef` *Calculate the number of simulated threat event frequencies (TEF)*

Description

Calculate the number of simulated threat event frequencies (TEF)

Usage

```
sample_tef(func = NULL, params = NULL)
```

Arguments

`func`	Function to use to simulate TEF, defaults to `rpert`.
`params`	Optional parameters to pass to `func`.

Value

List containing type ("tef"), samples (as a vector), and details (as a list).

See Also

Other OpenFAIR helpers: compare_tef_vuln, get_mean_control_strength, openfair_tef_tc_diff_lm, sample_diff, sample_lef, sample_lm, sample_tc, sample_vuln, select_loss_opportunities

---

`sample_vuln` *Calculate the vulnerability*

Description

Calculate the vulnerability

Usage

```
sample_vuln(func = NULL, params = NULL)
```

Arguments

`func`	Function to use to simulate VULN, defaults to `rbinom`.
`params`	Optional parameters to pass to `func`.

Value

List containing type ("vuln"), samples (as a vector), and details (as a list).

See Also

Other OpenFAIR helpers: `compare_tef_vuln`, `get_mean_control_strength`, `openfair_tef_tc_diff_lm`, `sample_diff`, `sample_lef`, `sample_lm`, `sample_tc`, `sample_tef`, `select_loss_opportunities`

---

scenario_summary *Scenario-level risk summary*

Description

A dataset of quantified information security risk, summarized at the scenario level.

Usage

```
scenario_summary
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 56 rows and 18 columns.

Details

domain_id domain id

scenario_id ID of the scenario

loss_events_mean mean number of loss events

loss_events_min minimum number of loss events

loss_events_max maximum number of loss events

loss_events_median median number of loss events

ale_median median annual loss expected

ale_max maximum annual loss expected

ale_var value at risk, ale

sle_mean mean single loss expectancy

sle_median median single loss expectancy

sle_max maximum single loss expectancy

sle_min minimum single loss expectancy

mean_tc_exceedance mean threat capability exceedance

mean_diff_exceedance mean difficulty exceedance

mean_vuln mean vulnerability of the scenario

ale_var_zscore Z-score of ale VaR

outlier boolean - is this scenario an outlier

select_loss_opportunities *Determine which threat events result in loss opportunities*

Description

Composition function for use in sample_vuln, does a simple compare of all threat events where the threat capability (TC) is greater than the difficulty (DIFF).

Usage

```
select_loss_opportunities(tc, diff)
```

Arguments

tc	Threat capability (as a percentage).
diff	Difficulty (as a percentage).

Value

List containing boolean values of length TC (as a vector) and details (as a list).

See Also

Other OpenFAIR helpers: compare_tef_vuln, get_mean_control_strength, openfair_tef_tc_diff_lm, sample_diff, sample_lef, sample_lm, sample_tc, sample_tef, sample_vuln

Examples

```
threat_capabilities <- c(.1, .5, .9)
difficulties <- c(.09, .6, .8)
select_loss_opportunities(threat_capabilities, difficulties)
```

simulation_results *Information security risk simulation results*

Description

A dataset containing the full results of sample Monte Carlo simulations of information security risk scenarios.

Usage

```
simulation_results
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 56000 rows and 13 columns.

Details

domain_id domain abbreviation

scenario_id id of the scenario

simulation id of the simulation

threat_events number of threat events

loss_events number of loss events occurring in the simulation

vuln percentage of threat events that result in losses

mean_tc_exceedance mean amount of TC > DIFF

mean_diff_exceedance mean amount of DIFF > TC

ale annual loss expectancy

sle_min single loss expectancy - minimum

sle_mean single loss expectancy - mean

sle_median single loss expectancy - mean

sle_max single loss expectancy - maximum

`split_sheet` *Split a sheet of the survey spreadsheet into either capabilities or threats*

Description

The default data collection Excel spreadsheet solicits threat scenarios and applicable controls for each domain. This function takes a single sheet from the spreadsheet, as read by `readxl` and pulls out either the capabilities or threats, as directed by the user.

Usage

```
split_sheet(dat, table_type = "capabilities")
```

Arguments

`dat`	Raw sheet input from `readxl`.
`table_type`	Either `capabilities` or `threats`

Value

Extracted table as a dataframe

summarize_domains *Create domain-level summary of simulation results*

Description

Given a dataframe of raw results from run_simulations, summarize the individual results at a per-domain level. This domain-level summary is a useful data structure for aggregate reporting.

Usage

```
summarize_domains(simulation_results)
```

Arguments

simulation_results	Simulation results dataframe.

Details

Summary stats created include:

- Mean/Min/Max/Median are calculated for loss events
- Median/Max/VaR are calculated for annual loss expected (ALE)
- Mean/Median/Max/Min are calculated for single loss expected (SLE)
- Mean percentage of threat capability exceeding difficulty on successful threat events
- Mean percentage of difficulty exceeding threat capability on defended events
- Vulnerability percentage
- Z-score of ALE (outliers flagged as 2 >= z-score)

Value

Simulation results summarized by domain

Examples

```
## Not run:
data(simulation_results)
summarize_domains(simulation_results)

## End(Not run)
```

summarize_scenarios *Create scenario-level summary of simulation results*

Description

Given a dataframe of raw results from run_simulations, summarize the individual results at a per-scenario level. This is generally the most granular level of data for reporting and analysis (full simulation results are rarely directly helpful).

Usage

```
summarize_scenarios(simulation_results)
```

Arguments

simulation_results	Simulation results dataframe.

Details

Summary stats created include: * Mean/Min/Max/Median are calculated for loss events * Median/Max/VaR are calculated for annual loss expected (ALE) * Mean/Median/Max/Min are calculated for single loss expected (SLE) * Mean percentage of threat capability exceeding difficulty on successful threat events * Mean percentage of difficulty exceeding threat capability on defended events * Vulnerability percentage * Z-score of ALE (outliers flagged as 2 >= z-score)

Value

Dataframe.

Examples

```
data(simulation_results)
summarize_scenarios(simulation_results)
```

summarize_simulations *Create simulation-level summary of simulation results*

Description

Given a dataframe of raw results from run_simulations, summarize the individual results at a per-simulation level.

Usage

```
summarize_simulations(simulation_results)
```

Arguments

simulation_results	Simulation results dataframe.

Details

Summary stats created include: * Mean/Min/Max/Median are calculated for loss events * Median/Max/VaR are calculated for annual loss expected (ALE) * Mean/Median/Max/Min are calculated for single loss expected (SLE) * Mean percentage of threat capability exceeding difficulty on successful threat events * Mean percentage of difficulty exceeding threat capability on defended events * Vulnerability percentage * Z-score of ALE (outliers flagged as 2 >= z-score)

Value

Dataframe.

Examples

```
data(simulation_results)
summarize_scenarios(simulation_results)
```

---

summarize_to_disk *Create all summary files and write to disk*

Description

This is a wrapper around summarize_scenarios and summarize_domains, calling both functions and writing the dataframes to a location on disk.

Usage

```
summarize_to_disk(simulation_results, domains, results_dir = "~/results")
```

Arguments

simulation_results	Simulation results dataframe.
domains	Domain mappings dataframe.
results_dir	Directory to place simulation files.

Value

Tibble with paths to the created data files.

Examples

```
## Not run:
data(simulation_results)
data(domains)
summarize_to_disk(simulation_results, domains)

## End(Not run)
```

---

theme_evaluator *Default ggplot theme used by all Evaluator-supplied graphics*

Description

Returns a standardized ggplot theme used by all built-in Evaluator plots.

Usage

```
theme_evaluator(base_family = "BentonSansRE")
```

Arguments

base_family	Font family.

Value

A ggplot theme object.

Examples

```
library(ggplot2)
p <- ggplot(mtcars) + geom_point(aes(wt, mpg, color = factor(gear))) + facet_wrap(~am)
font_family <- get_base_fontfamily()
p + theme_evaluator(font_family)
```

---

validate_scenarios *Validate qualitative scenario data*

Description

Run a set of basic consistency checks on the key qualitative data inputs (scenarios, capabilities, domains, and mappings).

Usage

```
validate_scenarios(scenarios, capabilities, domains, mappings)
```

Arguments

scenarios	Dataframe of qualitative scenarios.
capabilities	Dataframe of capabilities.
domains	Dataframe of domain mappings.
mappings	Dataframe of qualitative to quantitative mappings.

Details

Checks that:

- All scenarios are distinct
- All controls referenced in scenarios are defined in the controls table
- All controls are distinct

Value

A invisible boolean as to success/failure of validation steps.

Examples

```
## Not run:
validate_scenarios(scenarios, capabilities, domains, mappings)

## End(Not run)
```

Index