Package: kwb.qmra (via r-universe)

August 27, 2024

Title QMRA (quantitative microbial risk assessment)

Version 0.3.0

Description QMRA for water supply systems.

License MIT + file LICENSE

URL http://github.com/KWB-R/kwb.qmra/

BugReports https://github.com/KWB-R/kwb.qmra/issues

Depends R (>= 4.0.0)

Imports dplyr (>= 1.0.2), EnvStats (>= 2.3.1), ggplot2 (>= 3.3.2), jsonlite (>= 1.7.2), kwb.utils (>= 0.7.0), magrittr (>= 1.5), plyr (>= 1.8.6), readr (>= 1.4.0), readxl (>= 1.3.1), rlang (>= 0.4.8), rmarkdown (>= 2.4), rvest (>= 0.3.6), sfsmisc (>= 1.1.7), shiny (>= 1.5.0), stringr (>= 1.4.0), tidyr (>= 1.1.2), xml2 (>= 1.3.2)

Suggests ggrepel (>= 0.8.2), knitr (>= 1.30), qmra.db (>= 0.10.0), remotes (>= 2.2.0), sessioninfo (>= 1.1.1), stringi (>= 1.5.3), testthat (>= 3.0.0), tibble (>= 3.0.4)

VignetteBuilder knitr

Remotes github::kwb-r/kwb.utils@v0.7.0, github::kwb-r/qmra.db@v0.10.0

Encoding UTF-8

LazyLoad yes

RoxygenNote 7.1.1

Config/testthat/edition 3

Repository https://kwb-r.r-universe.dev

RemoteUrl https://github.com/KWB-R/kwb.qmra

RemoteRef HEAD

RemoteSha 9e096057da37bf574626c60e9ad524dff0b1d89a

Contents

`backcalc_infectionRisk_perDay`
Backcalculate risk: infection (using optimise() function)

Description

Based on exposure days per year and target infection risk per year the acceptable daily infection risk is backcalculated

Usage

```
backcalc_infectionRisk_perDay(
  target_infectionRisk_perYear = 1/10000,
  exposures_daysPerYear = 1:365
)
```

Arguments

`target_infectionRisk_perYear`	target infection risk per per year (default: 1/10000)
`exposures_daysPerYear`	exposure days per year (default: 1 to 365)

Value

acceptable daily infection risk for given exposures per year and target infection risk per year

`calc_health_risk` *Risk calculation: health*

Description

Risk calculation: health

Usage

```
calc_health_risk(
  infectionRisk_perYear = 9.5 * 10^-4,
  infection_to_illness = 0.7,
  diseaseBurden_dalyPerCase = 1.5 * 10^-3,
  fraction_population = 1,
  target_dalyPerYearPerPerson = 1/1e+06
)
```

Arguments

`infectionRisk_perYear`	as retrieved by calc_infection_risk()$events$infectionRisk_perYear or user defined input (default: 9.5 * 10 ^ - 4 infections/year)
`infection_to_illness`	probability of illness given infection (default: 0.7, i.e. 70 percent illness probability giving infection)
`diseaseBurden_dalyPerCase`	disabled adjusted life years per case. Value depends on multiple parameters such as illness type (default: 1.5*10 ^ -3)
`fraction_population`	fraction of population at risk of getting the illness (default: 1, i.e. 100 percent of population can possibly get the illness, worst-case assumption assuming no prior immunization!)
`target_dalyPerYearPerPerson`	target disabled adjusted life years (DALY) per person per year (default: 1/1000000 DALY per per person per year, WHO standard)

Value

data frame with input parameters and calculated health risk outputs

See Also

`calc_infection_risk` for infection risk input

`calc_infection_risk` *Risk calculation: infection*

Description

Risk calculation: infection

Usage

```
calc_infection_risk(
  inflow_orgPerLitre = 10,
  treatment_logRemoval = 5.8,
  exposure_daysPerYear = 365,
  doseresponse_modelType = "dr.expo",
  waterConsumption_LitrePerDay = 1,
  target_infectionRisk_perYear = 1/10000,
  ...
)
```

Arguments

`inflow_orgPerLitre`	concentration of microbiological parameter in inflow to water treatment plant (default: 10 Org/L)
`treatment_logRemoval`	reduction of microbiological concentration in water treatment plant (default: 5.8 log)
`exposure_daysPerYear`	exposure days per year (default: 365)
`doseresponse_modelType`	dose response model to be used: "dr.expo" for exponential or "dr.betapoisson" for beta-poisson model
`waterConsumption_LitrePerDay`	daily water consumption (default: 1 L/day)
`target_infectionRisk_perYear`	NOT IMPLEMENTED YET!!!! target infection risk per per year. Only used if one of the following input parameters (i.e. "inflow_orgPerLitre", "treatment_logRemoval", "exposure_daysPerYear", "waterConsumption_LitrePerDay") is not defined (default: 1/10000).
`...`	additional parameters used for dose response modelling. Depends on used dose-response model

Value

list with input parameters and calculated infection risk

See Also

`dr.expo` for exponential or `dr.betapoisson` for beta-poisson dose-response model

---

`config_default_json` *config_dummy_json: dummy configuration for kwb.qmra::opencpu_simulate_risk() in JSON format*

Description

JSON input configuration generated with R script (data-raw/config_json.R)

Usage

```
data(config_dummy_json)
```

Format

A json list with all configuration parameters used by opencpu_simulate_risk, which are passed to simulate_risk

Examples

```
data("config_dummy_json")
config_dummy_json
```

`config_dummy_json` *config_default_json: default configuration developed by Christoph Sprenger for kwb.qmra::opencpu_simulate_risk() in JSON format*

Description

JSON input configuration generated with R script (data-raw/config_json.R)

Usage

```
data(config_default_json)
```

Format

A json list with all configuration parameters used by opencpu_simulate_risk, which are passed to simulate_risk

Examples

```
data("config_default_json")
config_default_json
```

`config_read` *Config: read configuration*

Description

Config: read configuration

Usage

```
config_read(
  confDir = system.file("extdata/configs/dummy", package = "kwb.qmra")
)
```

Arguments

`confDir`	directory to read configuration files (Default: system.file('extdata/configs/dummy', package = 'kwb.qmra')")

Value

stores configuration in R list structure

`config_write` *Config: create configuration*

Description

Config: create configuration

Usage

```
config_write(config, confName = "dummy", confDir = tempdir(), zipFiles = TRUE)
```

Arguments

`config`	a configuration as retrieved by config_read()
`confName`	name of configuration
`confDir`	directory to save the configuration files (Default: tempdir()")
`zipFiles`	should also zipfile be created in folder confDir (default: TRUE)

Value

writes configuration in confDir subfolder defined in confName

`config_write_dummy` *Config: create dummy configuration*

Description

Config: create dummy configuration

Usage

```
config_write_dummy(
  confDir = system.file("extdata/configs/dummy", package = "kwb.qmra")
)
```

Arguments

`confDir`	directory to save the dummy configuration files (Default: system.file('extdata/config', package = 'kwb.qmra')")

Value

writes dummy configuration in confDir

`create_random_distribution` *Create random distribution*

Description

Create random distribution

Usage

```
create_random_distribution(
  type = "uniform",
  number_of_repeatings = 1,
  number_of_events = 365,
  value = 10,
  min = 10,
  max = 1000,
  percent_within_minmax = 0.9,
  min_zero = 0.01,
  log10_min = default_min(min, max, min_zero, f = log10),
  log10_max = default_max(max, min_zero * 10, f = log10),
  log10_mean = (log10_min + log10_max)/2,
 log10_sdev = abs((log10_max - log10_mean)/get_percentile(percent_within_minmax)),
  mean = (default_min(min, max, min_zero) + default_max(max, 10 * min_zero))/2,
  sdev = abs((default_max(max, 10 * min_zero) -
    mean)/get_percentile(percent_within_minmax)),
  meanlog = mean(log(default_min(min, max, min_zero) + default_max(max, 10 *
    min_zero))/2),
 sdlog = abs(sd(c(default_min(min, max, min_zero, f = log), default_max(max, 10 *
    min_zero, f = log)))),
  mode = (default_min(min, max, min_zero) + default_max(max, 10 * min_zero))/2,
  debug = TRUE
)
```

Arguments

`type`	"uniform" calls runif(), "log10_uniform" calls 10^runif(number_of_events, log10_min, log10_max), "triangle" calls EnvStats::rtri(), "lognorm" calls rlnorm(), "norm" calls rnorm() and "log10_norm" calls 10^rnorm(number_of_events, mean = log10_mean, sdev = log10_sdev), (default: "uniform")
`number_of_repeatings`	how often should the random distribution with the same parameters be generated (default: 1)
`number_of_events`	number of events
`value`	constant value (no random number), gets repeated number_of_events times (if 'type' = 'value')

`min`	minimum value (default: 10), only used if 'type' is "runif" or "triangle"
`max`	maximum value (default: 1000), only used if 'type' is "runif" or "triangle"
`percent_within_minmax`	percent of data point within min/max (default: 0.9 i.e. 90 percent
`min_zero`	only used if 'type' is "log10_uniform" or "log10_norm", "norm" or "lognorm" and "min" value equal zero. In this case the zero is replaced by this value (default: 0.01), see also `default_min`
`log10_min`	minimum value (default: default_min(min, max, min_zero, f = log10)), only used if 'type' is "log10_uniform" or "log10_norm"
`log10_max`	maximum value (default: ifelse(max > 0, log10(max), log10_zero_threshold), only used if 'type' is "log10_uniform" or "log10_norm"
`log10_mean`	mean value (default: (log10_min + log10_max)/2), only used if 'type' is "log10_norm"
`log10_sdev`	standard deviation (default: abs((log10_max- log10_mean) / get_percentile(0.95)), only used if 'type' is "log10_norm"
`mean`	mean value (default: (default_min(min, max, min_zero) / default_max(max, 10*min_zero)) / 2), only used if 'type' is "norm"
`sdev`	standard deviation (default: abs((default_max(max, 10*min_zero) - mean) / get_percentile(0.95))), only used if 'type' is "norm"
`meanlog`	log mean value (default: mean(log((min + max) / 2))), only used if 'type' is "lognorm"
`sdlog`	standard deviation (default: abs(sd(c(default_min(min, max, min_zero, f = log))))), only used if 'type' is "lognorm"
`mode`	(default: default_min(min, max, min_zero) + default_max(max, 10 * min_zero) / 2), only used if 'type' is "triangle"
`debug`	print debug information (default: TRUE)

Value

list with parameters of user defined random distribution and corresponding values

See Also

for random triangle see `rtri`, for default min/max see `default_min`, `default_max` and `get_percentile`

`default_max` *Default Max*

Description

Default Max

Usage

```
default_max(org_max, new_max, f = c)
```

Arguments

`org_max`	original maximum value
`new_max`	new maximum value for replacement
`f`	function apply on "org_max" to transform to the correct dimension (e.g. "log" in case of "lognorm" or "log10" in case of log10_norm) (default: c())

Examples

```
default_max(org_max = 2, new_max = 0.01, f = log10)
default_max(org_max = 0, new_max = 0.01, f = log10)
default_max(org_max = 2, new_max = 0.01, f = log)
default_max(org_max = 0, new_max = 0.01, f = log)
```

`default_min` *Default Min*

Description

Default Min

Usage

```
default_min(org_min, org_max, new_min, f = c)
```

Arguments

`org_min`	original minimum value
`org_max`	original maximum value
`new_min`	new minimum value for replacement
`f`	function apply on org_min and new_min to transform to the correct dimension (e.g. "log" in case of "lognorm" or "log10" in case of log10_norm), (default: c())

Examples

```
default_min(org_min = 2, org_max = 100, new_min = 0.01, f = log10)
default_min(org_min = 0, org_max = 100, new_min = 0.01, f = log10)
default_min(org_min = 2, org_max = 100, new_min = 0.01, f = log)
default_min(org_min = 0, org_max = 100, new_min = 0.01, f = log)
```

distribution_repeater *Helper function: distributon repeater*

Description

Helper function: distributon repeater

Usage

```
distribution_repeater(
  number_of_repeatings = 10,
  number_of_events = 365,
  func,
  ...
)
```

Arguments

`number_of_repeatings`	how often should the random distribution with the same parameters be generated (default: 1)
`number_of_events`	number of events
`func`	distribution function to be repeated (e.g. runif, rlnorm, rnorm)
`...`	further parameters passed to func

Value

data.frame with columns repeatID, eventID and values

Examples

```
distribution_repeater(
  number_of_repeatings = 2,
  number_of_events = 10,
  func = runif,
  min = 1,
  max = 10
)
```

dr.betapoisson *Dose-response model: beta-poisson*

Description

Dose-response model: beta-poisson

Usage

```
dr.betapoisson(
  dose = sfsmisc::lseq(from = 1, to = 10^10, length = 1000),
  alpha = 0.328,
  N50 = 5430
)
```

Arguments

dose	vector of dose data (default: `sfsmisc::lseq(from = 0.1, to = 10^10, length = 1000)`)
alpha	alpha (default: 3.28E-01)
N50	N50 (default: 5.43E+03)

Value

tibble

dr.db_download *Download dose-response model database from QMRAwiki*

Description

Download dose-response model database from QMRAwiki

Usage

```
dr.db_download(fromInternet = FALSE)
```

Arguments

fromInternet	download from internet (default: FALSE), if FALSE import from local copy

Value

tibble for different microbial parameters

Source

http://qmrawiki.canr.msu.edu/index.php?title=Table_of_Recommended_Best-Fit_Parameters

`dr.db_model` *Generate table with different doses for dr.db_download()*

Description

Generate table with different doses for dr.db_download()

Usage

```
dr.db_model(
  dr.db = dr.db_download(),
  dose = sfsmisc::lseq(from = 1, to = 10^10, length = 1000)
)
```

Arguments

`dr.db`	as retrieved by dr.db_download(), default: dr.db_download()
`dose`	vector of dose data (default: `sfsmisc::lseq(from=0.1, to = 10^10,length = 1000))`

Value

tibble

`dr.expo` *Dose-response model: exponential*

Description

Dose-response model: exponential

Usage

```
dr.expo(dose = sfsmisc::lseq(from = 1, to = 10^10, length = 1000), k = 0.572)
```

Arguments

`dose`	vector of dose data (default: `sfsmisc::lseq(from = 0.1, to = 10^10, length = 1000))`
`k`	k-value (default: 5.72E-01)

Value

tibble

generate_random_values *Create random distribution based on configuration file*

Description

Create random distribution based on configuration file

Usage

```
generate_random_values(
  config,
  number_of_repeatings = 1,
  number_of_events,
  debug = TRUE
)
```

Arguments

`config`	as retrieved by config_read()
`number_of_repeatings`	how often should the random distribution with the same parameters be generated (default: 1)
`number_of_events`	number of events
`debug`	print debug information (default: TRUE)

Value

list random distributions based on configuration file

get_percentile *Helper function: get percentile*

Description

Helper function: get percentile

Usage

```
get_percentile(percent_within_minmax = 0.9)
```

Arguments

`percent_within_minmax`	percent of data point within min/max (default: 0.9 i.e. 90 percent)

Examples

```
get_percentile(0.9)
get_percentile(0.95)
```

`opencpu_config_read` *OpenCPU wrapper: import config from CSV and convert to JSON*

Description

OpenCPU wrapper: import config from CSV and convert to JSON

Usage

```
opencpu_config_read(
  confDir = system.file("extdata/configs/dummy", package = "kwb.qmra")
)
```

Arguments

`confDir`	directory to read configuration files (Default: system.file('extdata/configs/dummy', package = 'kwb.qmra')")

Value

stores configuration in JSON format

Examples

```
### Example json config file
config_json <- kwb.qmra::opencpu_config_read()
head(config_json)
```

`opencpu_simulate_risk` *OpenCPU wrapper: run risk calculation and convert results to JSON format*

Description

OpenCPU wrapper: run risk calculation and convert results to JSON format

Usage

```
opencpu_simulate_risk(
  config = config_read(),
  usePoisson = TRUE,
  debug = TRUE,
  lean = TRUE
)
```

Arguments

`config`	config object as retrieved by `kwb.qmra::config_read()`
`usePoisson`	should a poisson proccess (see function dose_perEvent()) be used to calculate the dose_perEvent (TRUE) or just the exposure_perEvent column (FALSE), (default: TRUE)
`debug`	print debug information (default: TRUE)
`lean`	if `TRUE`, a "lean" version of this function is called, see `kwb.qmra:::simulate_risk_lean`, (default: TRUE)

Value

JSON list with parameters of user defined random distribution and corresponding values

Examples

```
### Example simulation run
## Read from JSON
config_json <- kwb.qmra::opencpu_config_read()
config <- jsonlite::fromJSON(config_json)
## Optionally directly import from CSVs
# config <- kwb.qmra::config_read()
risk <- kwb.qmra::opencpu_simulate_risk(config)
risk_json <- jsonlite::toJSON(risk, pretty = TRUE)
writeLines(text = risk_json, "risk.json")
```

`plot_doseresponse` *plot reduction*

Description

plotting

Usage

```
plot_doseresponse(risk)
```

Arguments

`risk`	list as retrieved by simulate_risk()

Value

ggplot for reduction

`plot_effluent` *plot effluent*

Description

plotting

Usage

```
plot_effluent(risk)
```

Arguments

`risk`	list as retrieved by simulate_risk()

Value

ggplot for effluent

`plot_event_dalys` *plot dalys_per_event*

Description

plotting

Usage

```
plot_event_dalys(risk)
```

Arguments

`risk`	list as retrieved by simulate_risk()

Value

ggplot for dalys_per_event

`plot_event_dose` *plot dose per event*

Description

plotting

Usage

```
plot_event_dose(risk)
```

Arguments

`risk`	list as retrieved by simulate_risk()

Value

ggplot for dose per event

`plot_event_exposure` *plot exposure per event*

Description

plotting

Usage

```
plot_event_exposure(risk)
```

Arguments

`risk`	list as retrieved by simulate_risk()

Value

ggplot for exposure per event

`plot_event_illnessProb` *plot illness probability*

Description

plotting

Usage

```
plot_event_illnessProb(risk)
```

Arguments

`risk`	list as retrieved by simulate_risk()

Value

ggplot for illness probability

`plot_event_infectionProb` *plot infection probability*

Description

plotting

Usage

```
plot_event_infectionProb(risk)
```

Arguments

`risk`	list as retrieved by simulate_risk()

Value

ggplot for infection probability

`plot_event_volume` *plot volume per event*

Description

plotting volume

Usage

```
plot_event_volume(risk)
```

Arguments

`risk`	list as retrieved by simulate_risk()

Value

ggplot for volume per event

`plot_inflow` *plot inflow*

Description

plotting

Usage

```
plot_inflow(risk)
```

Arguments

`risk`	list as retrieved by simulate_risk

Value

ggplot for inflow

`plot_reduction` *plot reduction*

Description

plotting

Usage

```
plot_reduction(risk)
```

Arguments

`risk`	list as retrieved by simulate_risk()

Value

ggplot for reduction

`plot_total_dalys` *plot total DALYs*

Description

plotting

Usage

```
plot_total_dalys(risk, labelling = FALSE, title = "", tolerance = 1e-06)
```

Arguments

`risk`	list as retrieved by simulate_risk()
`labelling`	if TRUE labels with absolute DALYs will be plotted (default: FALSE)
`title`	title for plot (default: "")
`tolerance`	accecptable tolerance level of risk (default: 1E-6)

Value

ggplot for total DALYs

plot_total_illnessProb

plot total illness probability

Description

plotting

Usage

```
plot_total_illnessProb(risk, tolerance = 1e-04)
```

Arguments

`risk`	list as retrieved by simulate_risk()
`tolerance`	accecptable tolerance level of risk (default: 1E-4)

Value

ggplot for total illness probability

plot_total_infectionProb

plot total infection probability

Description

plotting

Usage

```
plot_total_infectionProb(risk, tolerance = 1e-04)
```

Arguments

`risk`	list as retrieved by simulate_risk()
`tolerance`	accecptable tolerance level of risk (default: 1E-4)

Value

ggplot for total infection probability

`poisson_dose` *Helper function: poisson distribution based on exposure per event*

Description

Helper function: poisson distribution based on exposure per event

Usage

```
poisson_dose(exposure_perEvent)
```

Arguments

`exposure_perEvent`	exposed organisms per event

Value

dose per event based on poisson process

`report_workflow` *Create report (not working for "shiny" reports)*

Description

Create report (not working for "shiny" reports)

Usage

```
report_workflow(
  confDirs = system.file("extdata/configs/", package = "kwb.qmra"),
  report_template_dir = system.file("extdata/report", package = "kwb.qmra"),
  report_template_name = "workflow.Rmd",
  report_output_dir = NULL,
  openReport = TRUE
)
```

Arguments

`confDirs`	directory containing subdirectory/ies of QMRA configurations default: system.file("extdata/configs/", package = "kwb.qmra")
`report_template_dir`	report template directory (default: system.file("extdata/report", package = "kwb.qmra"))
`report_template_name`	default: "workflow.Rmd"

`report_output_dir`	directory where report should be saved, if NULL report_template_dir is used (default: NULL)
`openReport`	open report in browser default: TRUE

Value

generate html report

`risk_dummy_json` *risk_dummy_json: example risk_json object returned by kwb.qmra::opencpu_simulate_risk()*

Description

JSON risk result object generated with R script (data-raw/risk_json.R)

Usage

```
data(risk_dummy_json)
```

Format

A json list with all risk results generated by opencpu_simulate_risk, using the dataset ‘config_dummy_json‘

Examples

```
data("risk_dummy_json")
risk <- jsonlite::fromJSON(risk_dummy_json)
### only show "stats" elements (skip "events" and "total" due to much data)
risk_stats <- list(stats_total = risk$stats_total,
                   stats_logremoval = risk$stats_logremoval)
jsonlite::toJSON(risk_stats, pretty = TRUE)
```

`run_app` *Run shiny app*

Description

Run shiny app

Usage

```
run_app(
  appDir = system.file("extdata/shiny", package = "kwb.qmra"),
  launch.browser = TRUE,
  ...
)
```

Arguments

`appDir`	directory of shiny app (default: system.file("extdata/shiny", package = "kwb.qmra"))
`launch.browser`	If true, the system's default web browser will be launched automatically after the app is started. Defaults to true in interactive sessions only. This value of this parameter can also be a function to call with the application's URL.
`...`	additional parameters passed to shiny::runApp

`simulate_exposure` *Simulate: exposure*

Description

Simulate: exposure

Usage

```
simulate_exposure(config, debug = TRUE)
```

Arguments

`config`	as retrieved by config_read()
`debug`	print debug information (default: TRUE)

Value

list with parameters of user defined exposure scenario (number of events and volumes per Event)

`simulate_inflow` *Simulate: inflow*

Description

Simulate: inflow

Usage

```
simulate_inflow(config, debug = TRUE, lean = FALSE)
```

Arguments

`config`	as retrieved by config_read()
`debug`	print debug information (default: TRUE)
`lean`	if `TRUE`, only the "events" are returned (in a reduced version, i.e. without column `PathogenName` and with all ID columns being of class `integer`), otherwise a list with the events in element `events`" and the corresponding parameters in element `paras`. The default is `FALSE`, i.e. events and parameters are returned in a list.

Value

list with parameters of user defined random distribution and corresponding values

---

`simulate_risk` *Simulate: risk*

Description

Simulate: risk

Usage

```
simulate_risk(config, usePoisson = TRUE, debug = TRUE, lean = FALSE)
```

Arguments

`config`	as retrieved by config_read()
`usePoisson`	should a poisson proccess (see function dose_perEvent()) be used to calculate the dose_perEvent (TRUE) or just the exposure_perEvent column (FALSE), (default: TRUE)
`debug`	print debug information (default: TRUE)
`lean`	if `TRUE`, a "lean" version of this function is called, see `kwb.qmra:::simulate_risk_lean`

Value

list with parameters of user defined random distribution and corresponding values

---

`simulate_treatment` *Simulate: treatment*

Description

Simulate: treatment

Usage

```
simulate_treatment(config, wide = FALSE, debug = TRUE, lean = FALSE)
```

Arguments

`config`	as retrieved by config_read()
`wide`	if TRUE results will be converted to wide format (default: FALSE)
`debug`	print debug information (default: TRUE)
`lean`	lean (default: FALSE)

Value

list with parameters of user defined random distribution and corresponding values

---

`who_getTreatment` *Treatment: get WHO reductions*

Description

Treatment: get WHO reductions

Usage

```
who_getTreatment()
```

Value

list with $untidy and $tidy data frames

See Also

http://apps.who.int/iris/bitstream/10665/44584/1/9789241548151_eng.pdf#page=162

Index