Package ‘GeoTox’

January 20, 2025

Title Spatiotemporal Mixture Risk Assessment

Version 0.2.0

Description Connecting spatiotemporal exposure to individual and population-level risk via source-to-outcome continuum modeling. The package, methods, and case-studies are described in Messier, Reif, and Marvel (2024) <doi:10.1101/2024.09.23.24314096> and Eccles et al. (2023) <doi:10.1016/j.scitotenv.2022.158905>.

License MIT + file LICENSE

URL `https://niehs.github.io/GeoTox/`, `https://github.com/NIEHS/GeoTox`

Depends R (>= 4.4.0)

Imports dplyr, ggplot2, ggridges, rlang, sf, stats, stringr, tibble, tidyr, tidyselect, truncnorm, utils

Suggests ggpubr, httk, httr2, knitr, purrr, readr, readxl, rmarkdown, scales, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

BugReports `https://github.com/NIEHS/GeoTox/issues`

NeedsCompilation no

Author Skylar Marvel [aut] (<`https://orcid.org/0000-0002-2971-9743`>), David Reif [aut] (<`https://orcid.org/0000-0001-7815-6767`>), Kyle Messier [cre, aut] (<`https://orcid.org/0000-0001-9508-9623`>), Spatiotemporal Exposures and Toxicology Group [cph]

Maintainer Kyle Messier <`kyle.messier@nih.gov`>

Repository CRAN

Date/Publication 2024-11-15 14:20:02 UTC

Contents

`calculate_response` *Calculate response*

Description

Calculate mixture response for GeoTox population data

Usage

```
calculate_response(x, ...)
```

Arguments

`x`	GeoTox object
`...`	additional arguments passed to other functions. See details.

Details

Additional parameters include `time`, `BW`, and `scaling` for calc_internal_dose, and `max_mult` for calc_concentration_response.

Value

The same object with additional fields added or updated

See Also

calc_internal_dose, calc_invitro_concentration, calc_concentration_response

Examples

```
# Use a subset of the package data for demonstration purposes
set.seed(2357)
n <- 10 # Population size
m <- 5 # Number of regions
idx <- if (m < 100) sample(1:100, m) else 1:100

# Create GeoTox object and populate required fields
geoTox <- GeoTox() |>
  # Simulate populations for each region
  simulate_population(age = split(geo_tox_data$age, ~FIPS)[idx],
                      obesity = geo_tox_data$obesity[idx, ],
                      exposure = split(geo_tox_data$exposure, ~FIPS)[idx],
                      simulated_css = geo_tox_data$simulated_css,
                      n = n) |>
  # Estimated Hill parameters
  set_hill_params(geo_tox_data$dose_response |>
                    fit_hill(assay = "endp", chem = "casn") |>
                    dplyr::filter(!tp.sd.imputed, !logAC50.sd.imputed))

# Response computations can now be done
geoTox <- geoTox |> calculate_response()
```

`calc_concentration_response`

Calculate the mixture response from one of three different approaches: IA, GCA, or Hazard Quotient

---

Description

Calculate the combined response of multiple chemicals. It calculates the generalized concentration addition response, the independent action response, and a hazard quotient

Usage

```
calc_concentration_response(
  C_invitro,
  hill_params,
  max_mult = 1.5,
  fixed = FALSE
)
```

Arguments

`C_invitro`	in vitro concentrations
`hill_params`	output from `fit_hill()`
`max_mult`	upper bound multiplier for max response
`fixed`	if TRUE, sd = 0

Value

list of data frames

Examples

```
C_invitro <- list(
  matrix(1:8 / 1e3, ncol = 2, dimnames = list(NULL, c("c1", "c2"))),
  matrix(9:16 / 1e3, ncol = 2, dimnames = list(NULL, c("c1", "c2")))
)
hill_params <- fit_hill(
  data.frame(chem = rep(c("c1", "c2"), each = 3),
             logc = c(-1, 0, 1, 0, 1, 2),
             resp = c(10, 5, 0, 4, 2, 0) / 10),
  chem = "chem"
)

calc_concentration_response(C_invitro, hill_params)
calc_concentration_response(C_invitro, hill_params, fixed = TRUE)
```

`calc_independent_action` *Independent Action*

Description

Calculate independent action response for a set of chemicals with Hill concentration-response curves.

Usage

```
calc_independent_action(conc, max, AC50, Emax, n = 1)
```

Arguments

`conc`	concentrations in regular space
`max`	maximal (asymptotic) responses
`AC50`	concentrations of half-maximal response
`Emax`	maximum mixture response
`n`	Hill coefficients (slopes)

Details

The concentration is computed as:

$$IA = E_{max} \times \left(1 - \prod_i \left(1 - \frac{x_i}{E_{max}}\right)\right),$$

where $x_i = hill\_val(conc_i, max_i, AC50_i, n_i)$ is the Hill model response function for each chemical.

Value

response value

See Also

`hill_val`

Examples

```
n_chem <- 5
conc <- 10^sample(-1:4, n_chem, replace = TRUE)
max <- 80 * runif(n_chem)
AC50 <- 10^(5 * runif(n_chem) - 1)
Emax <- 100

calc_independent_action(conc, max, AC50, Emax)
```

`calc_internal_dose` *Calculate internal chemical dose*

Description

Estimate the internal dose from inhalation of a chemical given inhalation rate, time, and body weight

Usage

```
calc_internal_dose(C_ext, IR, time = 1, BW = 1, scaling = 1)
```

Arguments

`C_ext`	ambient chemical concentration in $\frac{mg}{m^3}$
`IR`	inhalation rate in $\frac{m^3}{day}$
`time`	total time in $days$
`BW`	body weight in kg
`scaling`	scaling factor encompassing any required unit adjustments

Details

Input `C_ext` must be a matrix or list of matrices. Input `IR` must be an atomic vector or list of atomic vectors. The `time`, `BW` and `scaling` arguments are scalars.

The internal dose is calculated as:

$$D_{int} = \frac{C_{ext} \times IR \times time}{BW} \times scaling$$

Value

list of matrices containing internal chemical doses in $\frac{mg}{kg}$

Examples

```
# Single population
C_ext <- matrix(1:15, ncol = 3)
IR <- 1:5
calc_internal_dose(C_ext, IR)

# Multiple populations
C_ext <- list(
  "a" = matrix(1:15 / 10, ncol = 3),
  "b" = matrix(1:8, ncol = 2)
)
IR <- list(1:5, 1:4 / 2)
calc_internal_dose(C_ext, IR)
```

calc_invitro_concentration

Calculate in vitro *concentration*

Description

Estimate the *in vitro* equivalent plasma concentration given internal chemical dose and steady-state plasma concentration.

Usage

```
calc_invitro_concentration(D_int, C_ss = NULL)
```

Arguments

`D_int`	internal chemical dose in $\frac{mg}{kg}$
`C_ss`	steady-state plasma concentration in $\frac{\mu M}{mg/kg}$

Details

Input `D_int` must be a matrix or list of matrices. Input `C_ss` must be a numeric atomic vector or matrix, or a list of those types.

The *in vitro* equivalent plasma concentration is calculated as:

$$C_{plasma} = C_{ss} \times D_{int}$$

Value

list of matrices containing concentrations in μM

Examples

```
# Single population
D_int <- matrix(1:15, ncol = 3)
C_ss <- 1:5
calc_invitro_concentration(D_int, C_ss)

# Multiple populations
D_int <- list(
  "a" = matrix(1:15 / 10, ncol = 3),
  "b" = matrix(1:8, ncol = 2)
)
C_ss <- list(1:5, 1:4 / 2)
calc_invitro_concentration(D_int, C_ss)
```

compute_sensitivity *Compute response sensitivity to parameter variation.*

Description

Compute response sensitivity to parameter variation.

Usage

```
compute_sensitivity(
  x,
  vary = c("age", "obesity", "css_params", "fit_params", "C_ext"),
  max_mult = NULL
)
```

Arguments

`x`	GeoTox object.
`vary`	which parameter to vary.
`max_mult`	input for calc_concentration_response step.

Value

output from calc_concentration_response

Examples

```
# Use a subset of the package data for demonstration purposes
set.seed(2357)
n <- 10 # Population size
m <- 5 # Number of regions
idx <- if (m < 100) sample(1:100, m) else 1:100

# Create GeoTox object and populate required fields
geoTox <- GeoTox() |>
  # Simulate populations for each region
  simulate_population(age = split(geo_tox_data$age, ~FIPS)[idx],
                      obesity = geo_tox_data$obesity[idx, ],
                      exposure = split(geo_tox_data$exposure, ~FIPS)[idx],
                      simulated_css = geo_tox_data$simulated_css,
                      n = n) |>
  # Estimated Hill parameters
  set_hill_params(geo_tox_data$dose_response |>
                    fit_hill(assay = "endp", chem = "casn") |>
                    dplyr::filter(!tp.sd.imputed, !logAC50.sd.imputed))

# Sensitivity computations can now be done
age_resp <- geoTox |> compute_sensitivity()
obesity_resp <- geoTox |> compute_sensitivity(vary = "obesity")
```

fit_hill — *Fit 2- or 3-parameter Hill model*

Description

Fit 2- or 3-parameter Hill model

Usage

```
fit_hill(
  x,
  conc = "logc",
  resp = "resp",
  fixed_slope = TRUE,
  chem = NULL,
  assay = NULL
)
```

Arguments

`x`	data frame of dose response data.
`conc`	column name of base-10 log scaled concentration.
`resp`	column name of response.
`fixed_slope`	if TRUE, slope is fixed at 1.
`chem`	(optional) column name of chemical identifiers.
`assay`	(optional) column name of assay identifiers.

Details

Optional `chem` and `assay` identifiers can be used to fit multiple chemicals and/or assays. Returned columns `tp` is the top asymptote and `logAC50` is the 50% response concentration. If the computation of the standard deviations of these two parameters fails, then the standard deviation is set equal to the parameter estimate and is indicated by the respective imputed flag being TRUE.

Value

data frame of fit parameters.

Examples

```
# Multiple assays, multiple chemicals
df <- geo_tox_data$dose_response
fit_hill(df, assay = "endp", chem = "casn")

# Single assay, multiple chemicals
df <- geo_tox_data$dose_response |>
  dplyr::filter(endp == "TOX21_H2AX_HTRF_CHO_Agonist_ratio")
```

```
fit_hill(df, chem = "casn")

# Single assay, single chemical
df <- geo_tox_data$dose_response |>
  dplyr::filter(endp == "TOX21_H2AX_HTRF_CHO_Agonist_ratio",
                casn == "510-15-6")
fit_hill(df)
# 3-parameter Hill model
fit_hill(df, fixed_slope = FALSE)
```

`GeoTox` *GeoTox S3 object*

Description

An S3 object that can be used to help organize the data and results of a GeoTox analysis.

Usage

```
GeoTox()

## S3 method for class 'GeoTox'
plot(x, type = c("resp", "hill", "exposure", "sensitivity"), ...)
```

Arguments

`x`	GeoTox object.
`type`	type of plot.
`...`	arguments passed to subsequent methods.

Value

a GeoTox S3 object

See Also

plot_resp, plot_hill, plot_exposure, plot_sensitivity

Examples

```
# Use a subset of the package data for demonstration purposes
set.seed(2357)
n <- 10 # Population size
m <- 5 # Number of regions
idx <- if (m < 100) sample(1:100, m) else 1:100

geoTox <- GeoTox() |>
  # Set region and group boundaries (for plotting)
  set_boundaries(region = geo_tox_data$boundaries$county,
                group  = geo_tox_data$boundaries$state) |>
  # Simulate populations for each region
  simulate_population(age           = split(geo_tox_data$age, ~FIPS)[idx],
                      obesity       = geo_tox_data$obesity[idx, ],
                      exposure      = split(geo_tox_data$exposure, ~FIPS)[idx],
                      simulated_css = geo_tox_data$simulated_css,
                      n             = n) |>
  # Estimated Hill parameters
  set_hill_params(geo_tox_data$dose_response |>
                    fit_hill(assay = "endp", chem = "casn") |>
                    dplyr::filter(!tp.sd.imputed, !logAC50.sd.imputed)) |>
  # Calculate response
  calculate_response() |>
  # Perform sensitivity analysis
  sensitivity_analysis()

# Print GeoTox object
geoTox

# Plot hill fits
plot(geoTox, type = "hill")
# Plot exposure data
plot(geoTox, type = "exposure", ncol = 5)
# Plot response data
plot(geoTox)
plot(geoTox, assays = "TOX21_H2AX_HTRF_CHO_Agonist_ratio")
# Plot sensitivity data
plot(geoTox, type = "sensitivity")
plot(geoTox, type = "sensitivity", assay = "TOX21_H2AX_HTRF_CHO_Agonist_ratio")
```

`geo_tox_data` *GeoTox Data*

Description

Sample data for use in vignettes and function examples. See the Package Data vignette, `vignette("package_data", package = "GeoTox")`, for details on how this data was gathered.

Usage

```
geo_tox_data
```

Format

A list with items:

exposure 2019 AirToxScreen exposure concentrations for a subset of chemicals in North Carolina counties.

dose_response Subset of chemicals curated by ICE cHTS as active within a set of assays.

age County population estimates for 7/1/2019 in North Carolina.

obesity CDC PLACES obesity data for North Carolina counties in 2020.

simulated_css Simulated steady-state plasma concentrations for various age groups and obesity status combinations.

boundaries County and state boundaries for North Carolina in 2019.

`get_fixed_age` *Get `C_ss` Data for Fixed Age*

Description

Get `C_ss` Data for Fixed Age

Usage

```
get_fixed_age(simulated_css, age)
```

Arguments

`simulated_css`	list of pre-generated `C_ss` data, for details see: `vignette("package_data", package = "GeoTox")`.
`age`	list of atomic vectors containing ages.

Value

list of matrices containing median `C_ss` values.

Examples

```
get_fixed_age(simulated_css = geo_tox_data$simulated_css,
              age = list(c(25, 35, 55), c(15, 60)))
```

`get_fixed_css` *Get Fixed `C_ss` Data*

Description

Get `C_ss` values for use in sensitivity_analysis and compute_sensitivity.

Usage

```
get_fixed_css(simulated_css, age, obesity, C_ss)
```

Arguments

`simulated_css`	list of pre-generated `C_ss` data, for details see: `vignette("package_data", package = "GeoTox")`.
`age`	list of atomic vectors containing ages.
`obesity`	list of atomic vectors containing obesity status.
`C_ss`	list of matrices containing `C_ss` values.

Value

list of matrices or atomic vectors containing `C_ss` values.

Examples

```
# Define inputs
age <- list(c(25, 35, 55),
            c(15, 60))
obesity <- list(c("Obese", "Normal", "Obese"),
                c("Normal", "Normal"))
C_ss <- sample_Css(simulated_css = geo_tox_data$simulated_css,
                   age = age,
                   obesity = obesity)

# Get fixed C_ss data
get_fixed_css(simulated_css = geo_tox_data$simulated_css,
              age = age,
              obesity = obesity,
              C_ss = C_ss)
```

`get_fixed_obesity` *Get `C_ss` Data for Fixed Obesity Status*

Description

Get `C_ss` Data for Fixed Obesity Status

Usage

```
get_fixed_obesity(simulated_css, obesity)
```

Arguments

`simulated_css`	list of pre-generated `C_ss` data, for details see: `vignette("package_data", package = "GeoTox")`.
`obesity`	list of atomic vectors containing obesity status.

Value

list of matrices containing median `C_ss` values.

Examples

```
get_fixed_obesity(simulated_css = geo_tox_data$simulated_css,
                  obesity = list(c("Obese", "Normal", "Obese"),
                                 c("Normal", "Normal")))
```

---

get_fixed_other *Get median* `C_ss` *Values*

Description

Get median `C_ss` Values

Usage

```
get_fixed_other(C_ss)
```

Arguments

`C_ss`	list of matrices containing `C_ss` data

Value

list of atomic vectors containing median `C_ss` values.

Examples

```
# Generate input C_ss data
age <- list(c(25, 35, 55),
            c(15, 60))
obesity <- list(c("Obese", "Normal", "Obese"),
                c("Normal", "Normal"))
C_ss <- sample_Css(simulated_css = geo_tox_data$simulated_css,
                   age = age,
                   obesity = obesity)

# Get median C_ss values
get_fixed_other(C_ss)
```

get_fixed_params *Get C_ss Data for Fixed C_ss Generation Parameters*

Description

Get C_ss Data for Fixed C_ss Generation Parameters

Usage

```
get_fixed_params(simulated_css, age)
```

Arguments

simulated_css	list of pre-generated C_ss data, for details see: vignette("package_data", package = "GeoTox").
age	list of atomic vectors containing ages.

Value

list of matrices containing C_ss values.

Examples

```
get_fixed_params(simulated_css = geo_tox_data$simulated_css,
                 age = list(c(25, 35, 55), c(15, 60)))
```

hill_conc *Hill model concentration*

Description

Calculate the concentration in regular space for a given response value.

Usage

```
hill_conc(resp, max, AC50, n)
```

Arguments

resp	response value
max	maximal (asymptotic) response
AC50	concentration of half-maximal response
n	Hill coefficient (slope)

Details

This is a regular space version of tcpl::tcplHillConc().

The concentration is computed as:

$$conc = AC50 * (\frac{max}{resp} - 1)^{-1/n}$$

Value

concentration in regular space

See Also

hill_val

Examples

```
hill_conc(c(0.2, 0.5, 0.75), 1, 0.01, 1)
hill_conc(c(0.2, 0.5, 0.9), 1, c(0.1, 0.01, 0.001), 2)
```

---

hill_val *Hill model response*

Description

Calculate the response for a given concentration in regular space.

Usage

```
hill_val(conc, max, AC50, n)
```

Arguments

conc	concentration in regular space
max	maximal (asymptotic) response
AC50	concentration of half-maximal response
n	Hill coefficient (slope)

Details

This is a regular space version of tcpl::tcplHillVal().

The Hill model is defined as:

$$resp = \frac{max}{1 + (\frac{AC50}{conc})^n}$$

Value

response value

See Also

hill_conc

Examples

```
hill_val(c(0.0025, 0.01, 0.03), 1, 0.01, 1)
hill_val(c(0.05, 0.01, 0.003), 1, c(0.1, 0.01, 0.001), 2)
```

plot_exposure *Plot exposure data.*

Description

Plot exposure data.

Usage

```
plot_exposure(
  exposure,
  region_boundary,
  group_boundary = NULL,
  chem_label = "chnm",
  ncol = 2
)
```

Arguments

exposure	list of exposure data named by region label.
region_boundary	"sf" data.frame mapping features to a "geometry" column. Used to color regions.
group_boundary	(optional) "sf" data.frame containing a "geometry" column. Used to draw outlines.
chem_label	label for facet_wrap.
ncol	number of columns to wrap.

Value

ggplot2 object.

Examples

```
# Load package data
exposure <- split(geo_tox_data$exposure, ~FIPS)
region_boundary <- geo_tox_data$boundaries$county
group_boundary <- geo_tox_data$boundaries$state

# Plot county exposure data
# Use CASN as label to avoid long chemical names
plot_exposure(exposure,
              region_boundary,
              chem_label = "casn",
              ncol = 5)

# Add state boundaries
plot_exposure(exposure,
              region_boundary,
              group_boundary = group_boundary,
              chem_label = "casn",
              ncol = 5)
```

---

plot_hill *Plot Hill equation fits.*

Description

Plot Hill equation fits.

Usage

```
plot_hill(hill_params, xlim = c(-1, 4))
```

Arguments

`hill_params`	output from `fit_hill`.
`xlim`	log-10 scaled concentration limits.

Value

ggplot2 object.

Examples

```
# Multiple assays, multiple chemicals
df <- geo_tox_data$dose_response
plot_hill(fit_hill(df, assay = "endp", chem = "casn"))

# Single assay, multiple chemicals
df <- geo_tox_data$dose_response |>
  dplyr::filter(endp == "TOX21_H2AX_HTRF_CHO_Agonist_ratio")
fig <- plot_hill(fit_hill(df, chem = "casn"))
fig
# Modify plot
fig + ggplot2::guides(color = ggplot2::guide_legend(title = "Chemical\nCASN"))

# Single assay, single chemical
df <- geo_tox_data$dose_response |>
```

```
  dplyr::filter(endp == "TOX21_H2AX_HTRF_CHO_Agonist_ratio",
                casn == "510-15-6")
plot_hill(fit_hill(df))
# 3-parameter Hill model
plot_hill(fit_hill(df, fixed_slope = FALSE))
```

`plot_resp` *Plot response data*

Description

Plot response data

Usage

```
plot_resp(
  df,
  region_boundary,
  group_boundary = NULL,
  assay_quantiles = c(Median = 0.5),
  summary_quantiles = c(`10th percentile` = 0.1)
)
```

Arguments

`df`	output from resp_quantiles.
`region_boundary`	"sf" data.frame mapping features to a "geometry" column. Used to color map regions.
`group_boundary`	"sf" data.frame containing a "geometry" column. Used to draw outlines around groups of regions.
`assay_quantiles`	named numeric vector of assay quantile labels.
`summary_quantiles`	named numeric vector of summary quantile labels.

Value

ggplot2 object.

Examples

```
# Use example boundary data from package
region_boundary <- geo_tox_data$boundaries$county
group_boundary <- geo_tox_data$boundaries$state
n <- nrow(region_boundary)
```

```
# Single assay quantile
df <- data.frame(id = region_boundary$FIPS,
                 metric = "GCA.Eff",
                 assay_quantile = 0.5,
                 value = runif(n)^3)
# Default plot
plot_resp(df, region_boundary)
# Add group boundary, a state border in this case
plot_resp(df, region_boundary, group_boundary)
# Change quantile label
plot_resp(df, region_boundary, group_boundary,
          assay_quantiles = c("Q50" = 0.5))

# Multiple assay quantiles
df <- data.frame(id = rep(region_boundary$FIPS, 2),
                 metric = "GCA.Eff",
                 assay_quantile = rep(c(0.25, 0.75), each = n),
                 value = c(runif(n)^3, runif(n)^3 + 0.15))
plot_resp(df, region_boundary, group_boundary,
          assay_quantiles = c("Q25" = 0.25, "Q75" = 0.75))

# Summary quantiles
df <- data.frame(id = rep(region_boundary$FIPS, 4),
                 assay_quantile = rep(rep(c(0.25, 0.75), each = n), 2),
                 summary_quantile = rep(c(0.05, 0.95), each = n * 2),
                 metric = "GCA.Eff",
                 value = c(runif(n)^3, runif(n)^3 + 0.15,
                           runif(n)^3 + 0.7, runif(n)^3 + 0.85))
plot_resp(df, region_boundary, group_boundary,
          assay_quantiles = c("A_Q25" = 0.25, "A_Q75" = 0.75),
          summary_quantiles = c("S_Q05" = 0.05, "S_Q95" = 0.95))
```

plot_sensitivity *Plot results of sensitivity analysis.*

Description

Plot results of sensitivity analysis.

Usage

```
plot_sensitivity(
  x,
  metric = "GCA.Eff",
  assay = NULL,
  y = "",
  xlab = metric,
  ylab = ""
)
```

Arguments

`x`	GeoTox object.
`metric`	metric to plot. Valid choices are "GCA.Eff", "IA.Eff", "GCA.HQ.10", and "IA.HQ.10".
`assay`	which assay to plot, if multiple exist.
`y`	y value or text for bottom of ridge plot.
`xlab`	x-axis label.
`ylab`	y-axis label.

Value

ggplot2 object.

Examples

```
# Required GeoTox fields are generated by first running [calculate_response]
# and [sensitivity_analysis] on a GeoTox object. This will create the fields
# `resp` and `sensitivity`. For this example, dummy data will be used.
make_data <- function(n = 5, metric = "GCA.Eff") {
  list(stats::setNames(data.frame(1:n, runif(n)),
                       c("sample", metric)))
}

geoTox <- GeoTox()
geoTox$resp <- make_data()
geoTox$sensitivity <- list(age = make_data(),
                           obesity = make_data(),
                           css_params = make_data(),
                           fit_params = make_data(),
                           C_ext = make_data())

plot_sensitivity(geoTox)
```

`resp_quantiles` *Get response quantiles*

Description

Get response quantiles

Usage

```
resp_quantiles(
  resp,
  metric = c("GCA.Eff", "IA.Eff", "GCA.HQ.10", "IA.HQ.10"),
  assays = NULL,
  assay_summary = FALSE,
  assay_quantiles = c(Median = 0.5),
  summary_quantiles = c(`10th percentile` = 0.1)
)
```

Arguments

`resp`	calculated mixture response output from calc_concentration_response.
`metric`	response metric, one of "GCA.Eff", "IA.Eff", "GCA.HQ.10" or "IA.HQ.10".
`assays`	assays to summarize. If NULL and multiple assays exist, then the first assay is used.
`assay_summary`	boolean indicating whether to summarize across assays.
`assay_quantiles`	numeric vector of assay quantiles.
`summary_quantiles`	numeric vector of quantiles to compute across all assay quantiles.

Details

The columns of the returned data frame will vary based on the inputs. If assays is specified and assay_summary is FALSE, then the resulting data frame will have an assay column. If assay_summary is TRUE, then the data frame will have an summary_quantile column.

Value

data frame with computed response quantiles.

Examples

```
# Dummy response data
resp <- list(
  "r1" = data.frame(assay = c("a1", "a1", "a2", "a2"),
                    sample = c(1, 2, 1, 2),
                    GCA.Eff = c(1, 2, 3, 4),
                    IA.Eff = c(5, 6, 7, 8),
                    "GCA.HQ.10" = c(9, 10, 11, 12),
                    "IA.HQ.10" = c(13, 14, 15, 16)))

# Summarize single assay
resp_quantiles(resp)
# Specify assay
resp_quantiles(resp, assays = "a1")
# Specify quantiles
resp_quantiles(resp, assays = "a1", assay_quantiles = c(0.25, 0.75))
# Specify metric
resp_quantiles(resp, assays = "a1", metric = "IA.HQ.10")

# Summarize across assays
resp_quantiles(resp, assay_summary = TRUE)
# Specify quantiles
suppressWarnings(
  resp_quantiles(resp,
                 assay_summary = TRUE,
                 assay_quantiles = c(0.25, 0.75),
                 summary_quantiles = c(0.1, 0.9))
)
```

`sample_Css` *Sample from pre-generated* `C_ss` *data*

Description

Sample from pre-generated `C_ss` data

Usage

```
sample_Css(simulated_css, age, obesity)
```

Arguments

`simulated_css`	list of pre-generated `C_ss` data, for details see: `vignette("package_data", package = "GeoTox")`.
`age`	list or atomic vector of ages.
`obesity`	list or atomic vector of obesity status.

Value

list of matrices containing `C_ss` values. Columns are sorted to have consistent order across functions.

Examples

```
# Vector inputs
sample_Css(geo_tox_data$simulated_css,
           c(15, 25, 35),
           c("Normal", "Obese", "Normal"))

# List inputs
sample_Css(geo_tox_data$simulated_css,
           list(c(34, 29), 55),
           list(c("Obese", "Normal"), "Normal"))
```

sensitivity_analysis *Perform sensitivity analysis*

Description

Perform sensitivity analysis

Usage

```
sensitivity_analysis(x, max_mult = list(NULL, NULL, NULL, 1.2, NULL))
```

Arguments

`x`	GeoTox object.
`max_mult`	numeric list of length 5 for each step of the sensitivity analysis.

Details

This wrapper function will sequentially call the compute_sensitivity function with inputs `age`, `obesity`, `css_params`, `fit_params`, and `C_ext`. The results will be returned as a named list and stored in the `sensitivity` field of the input GeoTox object.

Values of `NULL` in the `max_mult` input will use the default value stored in the `GeoTox` object (`x$par$resp$max_mult`). When a `GeoTox` object is created this is initialized at `1.5`, but can be changed via the calculate_response function or directly in the object.

Value

The same GeoTox object with added `sensitivity` field.

See Also

`compute_sensitivity`

Examples

```
# Use a subset of the package data for demonstration purposes
set.seed(2357)
n <- 10 # Population size
m <- 5 # Number of regions
idx <- if (m < 100) sample(1:100, m) else 1:100

# Create GeoTox object and populate required fields
geoTox <- GeoTox() |>
  # Simulate populations for each region
  simulate_population(age = split(geo_tox_data$age, ~FIPS)[idx],
                      obesity = geo_tox_data$obesity[idx, ],
                      exposure = split(geo_tox_data$exposure, ~FIPS)[idx],
                      simulated_css = geo_tox_data$simulated_css,
                     n = n) |>
  # Estimated Hill parameters
  set_hill_params(geo_tox_data$dose_response |>
                    fit_hill(assay = "endp", chem = "casn") |>
                    dplyr::filter(!tp.sd.imputed, !logAC50.sd.imputed))

# Sensitivity analysis can now be done
geoTox <- geoTox |> sensitivity_analysis()
```

`set_boundaries` *Set GeoTox boundaries*

Description

Set GeoTox boundaries

Usage

```
set_boundaries(x, region = NULL, group = NULL)
```

Arguments

`x`	GeoTox object.
`region`	"sf" data.frame mapping features to a "geometry" column. Used when coloring map regions.
`group`	"sf" data.frame containing a "geometry" column. Used to draw outlines around groups of regions.

Value

same GeoTox object with boundaries set.

Examples

```
geoTox <- GeoTox() |>
  set_boundaries(region = geo_tox_data$boundaries$county,
                 group  = geo_tox_data$boundaries$state)
```

set_hill_params — *Set Hill parameters for a GeoTox object.*

Description

Set Hill parameters for a GeoTox object.

Usage

```
set_hill_params(x, hill_params)
```

Arguments

`x`	GeoTox object.
`hill_params`	output of fit_hill.

Value

same GeoTox object with Hill parameters set.

Examples

```
hill_params <- geo_tox_data$dose_response |>
  fit_hill(chem = "casn", assay = "endp") |>
  dplyr::filter(!tp.sd.imputed, !logAC50.sd.imputed)

geoTox <- GeoTox() |>
  set_hill_params(hill_params)
```

simulate_age — *Simulate ages*

Description

Simulate ages

Usage

```
simulate_age(x, n = 1000)
```

Arguments

`x`	data frame or list of data frames containing population data for age groups. Each data frame must contain columns "AGEGRP" and "TOT_POP".
`n`	simulated sample size.

Details

Each data frame must contain 19 rows. The first row represents the total population of all age groups while the next 18 rows represent age groups from 0 to 89 in increments of 5 years.

Value

List of arrays containing simulated ages.

Examples

```
# Single data frame
x <- data.frame(AGEGRP = 0:18, TOT_POP = 0)
# populate only age range 40-44, set population total of all ages
x$TOT_POP[c(1, 10)] <- 100
simulate_age(x, 5)

# List of 2 data frames
y <- data.frame(AGEGRP = 0:18, TOT_POP = 0)
# populate age ranges 5-9 and 50-54
y$TOT_POP[c(3, 12)] <- 10
# set population total for all age groups
y$TOT_POP[1] <- sum(y$TOT_POP)
simulate_age(list(x = x, y = y), 15)
```

`simulate_exposure` *Simulate external exposure*

Description

Simulate external exposure

Usage

```
simulate_exposure(
  x,
  expos_mean = "mean",
  expos_sd = "sd",
  expos_label = "casn",
  n = 1000
)
```

Arguments

`x`	data frame or list of data frames containing exposure data.
`expos_mean`	column name of mean values.
`expos_sd`	column name of standard deviations.
`expos_label`	column name of labeling term, required if `x` has more than one row.
`n`	simulated sample size.

Value

list of matrices containing inhalation rates. Matrix columns are named using the values in the `expos_label` column for more than one data frame row. Columns are sorted to have consistent order across functions.

Examples

```
# Single data frame
x <- data.frame(mean = 1:3, sd = (1:3) / 10, casn = letters[1:3])
simulate_exposure(x, n = 5)

# List of 2 data frames
y <- data.frame(mean = 4:6, sd = 0.1, casn = letters[1:3])
simulate_exposure(list(loc1 = x, loc2 = y), n = 5)

# Input has custom column names
z <- data.frame(ave = 1:3, stdev = (1:3) / 10, chnm = letters[1:3])
simulate_exposure(z,
                  expos_mean = "ave",
                  expos_sd = "stdev",
                  expos_label = "chnm",
                  n = 5)
```

`simulate_inhalation_rate` *Simulate inhalation rates*

Description

Simulate inhalation rates

Usage

```
simulate_inhalation_rate(x, IR_params = NULL)
```

Arguments

`x`	atomic vector or list of atomic vectors containing ages.
`IR_params`	(optional) data frame with columns "age", "mean" and "sd". See details for more information.

Details

The age column of the optional `IR_params` data frame should be in ascending order and represent the lower value of age groups for the corresponding mean and sd values. When not provided, the default values will come from Table 6.7 of EPA's 2011 Exposure Factors Handbook using the mean of male and female values.

Value

List of atomic vectors containing inhalation rates.

Examples

```
# Single atomic vector
ages <- sample(1:100, 6, replace = TRUE)
simulate_inhalation_rate(ages)

# List of atomic vectors
ages <- list(
  sample(1:100, 5, replace = TRUE),
  sample(1:100, 3, replace = TRUE)
)
simulate_inhalation_rate(ages)

# Custom IR_params
IR_params <- data.frame("age" = c(0, 20, 50),
                        "mean" = c(0.5, 0.3, 0.2),
                        "sd" = c(0.1, 0.06, 0.03))
simulate_inhalation_rate(c(15, 30, 65), IR_params)
```

---

simulate_obesity *Simulate obesity status*

Description

Simulate obesity status

Usage

```
simulate_obesity(
  x,
  obes_prev = "OBESITY_CrudePrev",
  obes_sd = "OBESITY_SD",
  obes_label = "FIPS",
  n = 1000
)
```

Arguments

x	data frame containing obesity data as a percentage from 0 to 100.
obes_prev	column name of prevalence.
obes_sd	column name of standard deviation.
obes_label	column name of labeling term, required if x has more than one row.
n	simulated sample size.

Value

List of arrays containing simulated obesity status.

Examples

```
# Input has default column names
df <- data.frame(OBESITY_CrudePrev = c(20, 50, 80),
                 OBESITY_SD = c(5, 5, 5),
                 FIPS = letters[1:3])
simulate_obesity(df, n = 5)

# Input has custom column names
df <- data.frame(prev = c(20, 50, 80),
                 sd = c(5, 5, 5),
                 label = letters[1:3])
simulate_obesity(df,
                 obes_prev = "prev",
                 obes_sd = "sd",
                 obes_label = "label",
                 n = 5)
```

---

simulate_population *Simulate population data*

Description

Simulate population data for given input fields

Usage

```
simulate_population(
  x,
  age = NULL,
  obesity = NULL,
  exposure = NULL,
  simulated_css = NULL,
  ...
)
```

Arguments

x	GeoTox object.
age	input x to function simulate_age. After simulating ages, the inhalation rate is subsequently calculated using simulate_inhalation_rate.
obesity	input x to function simulate_obesity.
exposure	input x to function simulate_exposure.
simulated_css	input simulated_css to functions sample_Css and get_fixed_css.
...	additional arguments passed to other functions. See details.

Details

Additional parameters include `n` for sample size, `IR_params` for simulate_inhalation_rate, `obes_prev`, `obes_sd`, and `obes_label` for simulate_obesity, and `expos_mean`, `expos_sd`, and `expos_label` for simulate_exposure.

Value

The same object with simulated fields added.

Examples

```
# Use a subset of the package data for demonstration purposes
set.seed(2357)
n <- 10 # Population size
m <- 5 # Number of regions
idx <- if (m < 100) sample(1:100, m) else 1:100

# Create GeoTox object
geoTox <- GeoTox() |>
  # Simulate populations for each region
  simulate_population(age = split(geo_tox_data$age, ~FIPS)[idx],
                      obesity = geo_tox_data$obesity[idx, ],
                      exposure = split(geo_tox_data$exposure, ~FIPS)[idx],
                      simulated_css = geo_tox_data$simulated_css,
                      n = n)
```

Index