# Package 'GeoTox'

May 19, 2026

**Title** Spatiotemporal Mixture Risk Assessment

**Version** 1.0.0

**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 (2025) <doi:10.1186/s40246-024-00711-8> 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

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

**Depends** R (>= 4.4.0)

**Imports** DBI, dbplyr, dplyr, duckdb, duckplyr, purrr, rlang, tibble, tidyr, tidyselect, truncnorm

**Suggests** blob, ggplot2, ggridges, httk, httr2, knitr, quarto, readr, readxl, rmarkdown, sf, stringr, testthat (>= 3.0.0), withr, zip

**VignetteBuilder** knitr, quarto

**Config/Needs/website** rmarkdown, quarto

**Config/roxygen2/version** 8.0.0

**Config/testthat/edition** 3

**Encoding** UTF-8

**LazyData** true

**NeedsCompilation** no

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

**Maintainer** Skylar Marvel <skylar.marvel@nih.gov>

**Repository** CRAN

**Date/Publication** 2026-05-19 21:20:18 UTC

# Contents



---

`add_age` *Add age simulation data*

---

## Description

Create or add to the 'age' table in a GeoTox database.

## Usage

```
add_age(GT, df, location = "FIPS")
```

## Arguments

| | |
|---|---|
| `GT` | GeoTox object. |
| `df` | Data frame with age simulation data. |
| `location` | Column name(s) in `df` that contain location identifier(s) (default "FIPS"). |

## Details

The simulation data must have columns "AGEGRP" and "TOT_POP" and at least one column containing location information (default "FIPS"). Each location must have 19 rows for AGEGRP 0-18, where 1-18 are age groups in increments of 5, e.g. AGEGRP = 5 would be ages 20 to 24, and AGEGRP = 0 is the combination of all age groups. The age data is used by `simulate_age()` to generate age samples for each location.

The `location` input can be a named vector to specify multiple identifier columns in `df`. For example, `location = c(FIPS = "FIPS", state = "ST")` would indicate that `df` contains both FIPS codes and state identifiers for locations. The `state = "ST"` part would rename the "ST" column in `df` to "state" in the 'location' table.

## Value

The same GeoTox object, invisibly.

## See Also

`simulate_age()`

## Examples

```
# Example age simulation data
age_df <- data.frame(
  FIPS = rep(c(10000, 20000), each = 19),
  AGEGRP = rep(0:18, times = 2),
  TOT_POP = 0
)
# FIPS 10000, populate age group 40-44
age_df$TOT_POP[c(1, 10)] = 100
# FIPS 20000, populate age groups 50-59
age_df$TOT_POP[c(1, 12, 13) + 19] = c(200, 100, 100)

# Add age simulation data to GeoTox database
GT <- GeoTox() |> add_age(age_df)

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at created tables

dplyr::tbl(con, "age") |> dplyr::filter(TOT_POP > 0) |> dplyr::collect()

dplyr::tbl(con, "location") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

add_exposure *Add exposure simulation data*

## Description

Create or add to the 'exposure' table in a GeoTox database.

## Usage

```
add_exposure(GT, df, location = "FIPS", substance = "casn", route = "route")
```

## Arguments

| | |
|---|---|
| GT | GeoTox object. |
| df | Data frame with exposure simulation data. |
| location | Column name(s) in df that contain location identifier(s) (default "FIPS"). |
| substance | Column name(s) in df that contain substance identifier(s) (default "casn"). |
| route | Column name in df that contains route identifier (default "route"). |

## Details

The simulation data must contain columns for exposure mean and standard deviation (default "mean" and "sd", respectfully), at least one column containing location information (default "FIPS"), at least one column containing substance information (default "casn"), and one column containing route information (default "route"). The exposure data is used by `simulate_exposure()` to generate external exposure concentration samples for each location.

The `location` and `substance` inputs can be named vectors to specify multiple identifier columns in `df`. For example, `substance = c(casn = "casn", name = "chnm")` would indicate that `df` contains both CAS numbers and chemical names for substances. The `name = "chnm"` part would rename the "chnm" column in `df` to "name" in the 'substance' table.

## Value

The same GeoTox object, invisibly.

## See Also

`simulate_exposure()`

## Examples

```
# Example exposure simulation data
exposure_df <- tibble::tribble(
  ~FIPS, ~casn, ~route, ~mean, ~sd,
  10000, "00-00-1", "inhalation", 10, 1,
  10000, "00-00-2", "inhalation", 20, 1,
  20000, "00-00-1", "inhalation", 30, 1,
```

```
  20000, "00-00-2", "inhalation", 40, 1
)

# Add exposure simulation data to GeoTox database
GT <- GeoTox() |> add_exposure(exposure_df)

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at created tables
dplyr::tbl(con, "exposure") |> dplyr::collect()
dplyr::tbl(con, "location") |> dplyr::collect()
dplyr::tbl(con, "substance") |> dplyr::collect()
dplyr::tbl(con, "route") |> dplyr::collect()

# Add another substance with a new field and route
exposure_df <- tibble::tribble(
  ~FIPS, ~casn, ~chnm, ~route, ~mean, ~sd,
  10000, "00-00-3", "chem3", "drinking", 100, 1,
  20000, "00-00-3", "chem3", "drinking", 200, 1
)
GT |> add_exposure(exposure_df, substance = c(casn = "casn", name = "chnm"))

# Look at updated tables

dplyr::tbl(con, "exposure") |> dplyr::collect()

dplyr::tbl(con, "substance") |> dplyr::collect()

dplyr::tbl(con, "route") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`add_exposure_rate_params` *Add exposure rate simulation data*

---

## Description

Create or add to the 'exposure_rate_params' table in a GeoTox database.

## Usage

```
add_exposure_rate_params(
  GT,
  route = "inhalation",
  params = NULL,
  overwrite = FALSE
)
```

## Arguments

| | |
|---|---|
| GT | GeoTox object. |
| route | Exposure route (default "inhalation"). |
| params | Data frame with exposure rate parameters. If NULL, default parameters for the specified route will be used (default NULL). |
| overwrite | Logical indicating whether to overwrite existing exposure rate parameters for the specified route (default FALSE). |

## Details

There are two routes with built-in default exposure rate parameters: "inhalation" and "drinking". If params is not provided (NULL), the default parameters for the specified route will be used. If params is provided, it must contain columns "age_lb", "age_ub", "mean", and "sd". The exposure rate parameters are used by simulate_exposure_rate() to generate exposure rate samples for each individual in the population.

The build-in parameters come from the following sources:

- Inhalation: EPA Exposure Factors Handbook (2015), Table 6.7 in $m^3$/kg-day. The "mean" and "sd" values are the average of male and female values.
- Drinking: EPA Exposure Factors Handbook (2019), Table 3-30 in mL/kg-day.

## Value

The same GeoTox object, invisibly.

## See Also

simulate_exposure_rate()

## Examples

```
# Add both default params to GeoTox database
GT <- GeoTox() |>
  add_exposure_rate_params() |>
  add_exposure_rate_params(route = "drinking")

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at relevant tables
params_tbl <- dplyr::tbl(con, "exposure_rate_params") |> dplyr::collect()

params_tbl |> dplyr::filter(route_id == 1)

params_tbl |> dplyr::filter(route_id == 2)

dplyr::tbl(con, "route") |> dplyr::collect()

# Add custom params with new column (gender), assign to route "custom"
```

```
params_df <- tibble::tribble(
  ~age_lb, ~age_ub, ~gender, ~mean, ~sd,
   0, 49, "male",   10, 1,
  50, 99, "male",   20, 1,
   0, 49, "female", 30, 1,
  50, 99, "female", 40, 1
)
GT |> add_exposure_rate_params(params_df, route = "custom")

# Look at updated tables
params_tbl <- dplyr::tbl(con, "exposure_rate_params") |> dplyr::collect()

params_tbl |> dplyr::filter(route_id == 1)

params_tbl |> dplyr::filter(route_id == 2)

params_tbl |> dplyr::filter(route_id == 3)

dplyr::tbl(con, "route") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`add_hill_params` *Add Hill model parameters*

---

## Description

Create or add to the 'hill_params' table in a GeoTox database.

## Usage

```
add_hill_params(GT, hill_params)
```

## Arguments

| | |
|---|---|
| `GT` | GeoTox object. |
| `hill_params` | List output from `fit_hill()`. |

## Value

The same GeoTox object, invisibly.

## See Also

`fit_hill()`

## Examples

```
# Example Hill model data
hill_df <- tibble::tribble(
  ~assay, ~model, ~casn, ~logc, ~resp,
  "a1", "human", "00-00-1",    0,  10,
  "a1", "human", "00-00-1",    1,  20,
  "a1", "human", "00-00-1",    2,  80,
  "a1", "human", "00-00-1",    3, 100,
  "a1", "human", "00-00-2", -0.5,   5,
  "a1", "human", "00-00-2",  0.5,  20,
  "a1", "human", "00-00-2",  1.5,  55,
  "a1", "human", "00-00-2",  2.5,  60,
  "a2",   "rat", "00-00-1",   -1,   0,
  "a2",   "rat", "00-00-1",    0,  10,
  "a2",   "rat", "00-00-1",    1,  30,
  "a2",   "rat", "00-00-1",    2,  40
)
hill_params <- fit_hill(
  hill_df, assay = c(name = "assay", model = "model"), substance = "casn"
)

# Add Hill model parameters to GeoTox database
GT <- GeoTox() |> add_hill_params(hill_params)

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at created tables

dplyr::tbl(con, "hill_params") |> dplyr::collect()

dplyr::tbl(con, "assay") |> dplyr::collect()

dplyr::tbl(con, "substance") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`add_obesity` *Add obesity simulation data*

---

## Description

Create or add to the 'obesity' table in a GeoTox database.

## Usage

```
add_obesity(GT, df, location = "FIPS")
```

## Arguments

| | |
|---|---|
| GT | GeoTox object. |
| df | Data frame with obesity simulation data. |
| location | Column name(s) in df that contain location identifier(s) (default "FIPS"). |

## Details

The simulation data must contain columns for obesity prevalence and standard deviation (default "OBESITY_CrudePrev" and "OBESITY_SD", respectfully) and at least one column containing location information (default "FIPS"). The obesity data is used by simulate_obesity() to generate weight category samples for each location.

The location input can be a named vector to specify multiple identifier columns in df. For example, location = c(FIPS = "FIPS", state = "ST") would indicate that df contains both FIPS codes and state identifiers for locations. The state = "ST" part would rename the "ST" column in df to "state" in the 'location' table.

## Value

The same GeoTox object, invisibly.

## See Also

simulate_obesity()

## Examples

```
# Example obesity simulation data
obesity_df <- data.frame(
  FIPS = c(10000, 20000),
  OBESITY_CrudePrev = c(20, 80),
  OBESITY_SD = 5
)

# Add obesity simulation data to GeoTox database
GT <- GeoTox() |> add_obesity(obesity_df)

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at created tables

dplyr::tbl(con, "obesity") |> dplyr::collect()

dplyr::tbl(con, "location") |> dplyr::collect()

# Add another location with additional information
obesity_df <- data.frame(
 FIPS = 30000,
 ST = "State3",
 OBESITY_CrudePrev = 50,
```

```
 OBESITY_SD = 10
)
GT |> add_obesity(obesity_df, location = c(FIPS = "FIPS", state = "ST"))

# Look at updated tables

dplyr::tbl(con, "obesity") |> dplyr::collect()

dplyr::tbl(con, "location") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`calc_internal_dose` *Calculate internal dose*

---

## Description

Calculates internal dose (D_int) as the product of external concentration ('C_ext' in the 'concentration' table) and exposure rate ('rate' in the 'exposure_rate' table), and stores the results in the 'D_int' column of the 'concentration' table in the GeoTox database.

## Usage

```
calc_internal_dose(GT, overwrite = FALSE, sensitivity = FALSE)
```

## Arguments

| | |
|---|---|
| `GT` | GeoTox object. |
| `overwrite` | Logical indicating whether to overwrite existing 'D_int' values in the 'concentration' table (default FALSE). |
| `sensitivity` | Logical indicating whether to simulate internal dose for sensitivity analysis (default FALSE). |

## Details

If `sensitivity = TRUE`, 'D_int' will be calculated for sensitivity analysis. Typically this shouldn't be used directly by the user, but rather called by `calc_sensitivity()`. In this case, the function will use the 'concentration_sensitivity' and 'exposure_rate_sensitivity' tables instead of the 'concentration' and 'exposure_rate' tables.

## Value

The same GeoTox object, invisibly.

## See Also

`calc_response()`

## Examples

```
# Setup required tables
sample_df <- tibble::tribble(
  ~FIPS, ~age, ~weight,
  10000, 25, "Normal",
  10000, 35,  "Obese",
  20000, 50, "Normal"
)
exposure_df <- tibble::tribble(
  ~FIPS, ~casn, ~route, ~mean, ~sd,
  10000, "00-00-1", "inhalation", 10, 1,
  10000, "00-00-2", "inhalation", 20, 1,
  20000, "00-00-1", "inhalation", 30, 1,
  20000, "00-00-2", "inhalation", 40, 1
)
GT <- GeoTox() |>
  set_sample(sample_df) |>
  add_exposure_rate_params() |>
  simulate_population(exposure = exposure_df, sample_css = FALSE)

# Calculate internal dose
GT <- GT |> calc_internal_dose()

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at relevant tables

dplyr::tbl(con, "concentration") |> dplyr::collect()

dplyr::tbl(con, "exposure_rate") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

# calc_invitro_concentration

*Calculate in vitro concentration*

---

## Description

Calculates the in vitro concentration (C_invitro) as the product of steady state plasma concentration (C_ss) and internal dose (D_int), and stores the results in the 'C_invitro' column of the 'concentration' table in the GeoTox database.

## Usage

```
calc_invitro_concentration(GT, overwrite = FALSE, sensitivity = FALSE)
```

## Arguments

| | |
|---|---|
| GT | GeoTox object. |
| overwrite | Logical indicating whether to overwrite existing 'C_invitro' values in the 'concentration' table (default FALSE). |
| sensitivity | Logical indicating whether to simulate in vitro concentration for sensitivity analysis (default FALSE). |

## Details

If sensitivity = TRUE, 'C_invitro' will be calculated for sensitivity analysis. Typically this shouldn't be used directly by the user, but rather called by calc_sensitivity(). In this case, the function will use the 'concentration_sensitivity' table instead of the 'concentration' table.

## Value

The same GeoTox object, invisibly.

## See Also

calc_internal_dose(), calc_response()

## Examples

```
# Setup required tables
sample_df <- tibble::tribble(
  ~FIPS, ~age, ~weight,
  10000, 25, "Normal",
  10000, 35,  "Obese",
  20000, 50, "Normal"
)
exposure_df <- tibble::tribble(
  ~FIPS, ~casn, ~route, ~mean, ~sd,
  10000, "00-00-1", "inhalation", 10, 1,
  10000, "00-00-2", "inhalation", 20, 1,
  20000, "00-00-1", "inhalation", 30, 1,
  20000, "00-00-2", "inhalation", 40, 1
)
css_df <- tibble::tribble(
  ~casn, ~age_lb, ~age_ub, ~weight, ~css,
  "00-00-1",  0, 49, "Normal", 21,
  "00-00-1", 50, 99, "Normal", 22,
  "00-00-1",  0, 49,  "Obese", 61,
  "00-00-1", 50, 99,  "Obese", 62,
  "00-00-2",  0, 49, "Normal", 11,
  "00-00-2", 50, 99, "Normal", 12,
  "00-00-2",  0, 49,  "Obese", 31,
  "00-00-2", 50, 99,  "Obese", 32
)
GT <- GeoTox() |>
  set_sample(sample_df) |>
  set_simulated_css(css_df) |>
```

```
  add_exposure_rate_params() |>
  simulate_population(exposure = exposure_df) |>
  calc_internal_dose()

# Calculate in vitro concentration
GT <- GT |> calc_invitro_concentration()

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at relevant tables
dplyr::tbl(con, "concentration") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`calc_response` *Calculate concentrations and risks*

---

## Description

Calculate internal dose, in vitro concentration, and risk estimates.

## Usage

```
calc_response(GT, overwrite = FALSE, ...)
```

## Arguments

| | |
|---|---|
| `GT` | GeoTox object. |
| `overwrite` | Logical indicating whether to overwrite existing values (default FALSE). |
| `...` | Additional arguments passed to `calc_risk()`. |

## Details

This is a wrapper around several other functions:

- `calc_internal_dose()` to calculate internal dose (D_int).
- `calc_invitro_concentration()` to calculate in vitro concentration (C_invitro).
- `calc_risk()` to calculate risk estimates.

If a `risk_name` argument is provided to `...` and it is not "risk", then sensitivity analysis is assumed and the `sensitivity` argument in the internal dose and in vitro concentration calculations is set to TRUE.

## Value

The same GeoTox object, invisibly.

## See Also

calc_internal_dose(), calc_invitro_concentration(), calc_risk()

## Examples

```
# Setup required tables
sample_df <- tibble::tribble(
  ~FIPS, ~age, ~weight,
  10000, 25, "Normal",
  10000, 35,  "Obese",
  20000, 50, "Normal"
)
exposure_df <- tibble::tribble(
  ~FIPS, ~casn, ~route, ~mean, ~sd,
  10000, "00-00-1", "inhalation", 10, 1,
  10000, "00-00-2", "inhalation", 20, 1,
  20000, "00-00-1", "inhalation", 30, 1,
  20000, "00-00-2", "inhalation", 40, 1
)
css_df <- tibble::tribble(
  ~casn, ~age_lb, ~age_ub, ~weight, ~css,
  "00-00-1",  0, 49, "Normal", 21,
  "00-00-1", 50, 99, "Normal", 22,
  "00-00-1",  0, 49,  "Obese", 61,
  "00-00-1", 50, 99,  "Obese", 62,
  "00-00-2",  0, 49, "Normal", 11,
  "00-00-2", 50, 99, "Normal", 12,
  "00-00-2",  0, 49,  "Obese", 31,
  "00-00-2", 50, 99,  "Obese", 32
)
hill_df <- tibble::tribble(
  ~assay, ~casn, ~logc, ~resp,
  "a1", "00-00-1",    0,  10,
  "a1", "00-00-1",    1,  20,
  "a1", "00-00-1",    2,  80,
  "a1", "00-00-1",    3, 100,
  "a1", "00-00-2", -0.5,   5,
  "a1", "00-00-2",  0.5,  20,
  "a1", "00-00-2",  1.5,  55,
  "a1", "00-00-2",  2.5,  60
)
GT <- GeoTox() |>
  set_sample(sample_df) |>
  set_simulated_css(css_df) |>
  add_exposure_rate_params() |>
  add_hill_params(fit_hill(hill_df, assay = "assay", substance = "casn")) |>
  simulate_population(exposure = exposure_df)

# Calculate concentrations and risk
GT <- GT |> calc_response()

# Open a connection to GeoTox database
```

```
con <- get_con(GT)

# Look at relevant table

dplyr::tbl(con, "concentration") |> dplyr::collect()

dplyr::tbl(con, "risk") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`calc_risk` *Calculate risk scores*

---

## Description

Calculate generalized concentration addition (GCA) and independent action (IA) risk scores and hazard quotients (HQ) based on in vitro concentration data and Hill parameters.

## Usage

```
calc_risk(
  GT,
  max_mult = 1.5,
  fixed = FALSE,
  overwrite = FALSE,
  risk_name = "risk"
)
```

## Arguments

| | |
|---|---|
| `GT` | GeoTox object. |
| `max_mult` | Upper bound multiplier for max response (default 1.5). |
| `fixed` | Logical indicating whether to set standard deviation parameters of Hill fit to zero (default FALSE). Used in sensitivity analysis. |
| `overwrite` | Logical indicating whether to overwrite existing risk table (default FALSE). |
| `risk_name` | Table name to store risk results (default "risk"). Values other than "risk" are used in sensitivity analysis. |

## Details

This function requires that the 'concentration', 'sample', and 'hill_params' tables are present in the GeoTox object. Typically these tables will have been created by prior calls to `calc_invitro_concentration()` and `add_hill_params()`.

The risk scores are calculated for each sample and assay combination and stored in the 'risk' table in the GeoTox object, unless a different `risk_name` is provided. Supplying a different `risk_name` shouldn't be done directly by the user, but rather by calling `calc_sensitivity()`.

## Value

The same GeoTox object, invisibly.

## See Also

add_hill_params(), calc_invitro_concentration(), calc_response()

## Examples

```
# Setup required tables
sample_df <- tibble::tribble(
  ~FIPS, ~age, ~weight,
  10000, 25, "Normal",
  10000, 35,  "Obese",
  20000, 50, "Normal"
)
exposure_df <- tibble::tribble(
  ~FIPS, ~casn, ~route, ~mean, ~sd,
  10000, "00-00-1", "inhalation", 10, 1,
  10000, "00-00-2", "inhalation", 20, 1,
  20000, "00-00-1", "inhalation", 30, 1,
  20000, "00-00-2", "inhalation", 40, 1
)
css_df <- tibble::tribble(
  ~casn, ~age_lb, ~age_ub, ~weight, ~css,
  "00-00-1",  0, 49, "Normal", 21,
  "00-00-1", 50, 99, "Normal", 22,
  "00-00-1",  0, 49,  "Obese", 61,
  "00-00-1", 50, 99,  "Obese", 62,
  "00-00-2",  0, 49, "Normal", 11,
  "00-00-2", 50, 99, "Normal", 12,
  "00-00-2",  0, 49,  "Obese", 31,
  "00-00-2", 50, 99,  "Obese", 32
)
hill_df <- tibble::tribble(
  ~assay, ~casn, ~logc, ~resp,
  "a1", "00-00-1",    0,  10,
  "a1", "00-00-1",    1,  20,
  "a1", "00-00-1",    2,  80,
  "a1", "00-00-1",    3, 100,
  "a1", "00-00-2", -0.5,   5,
  "a1", "00-00-2",  0.5,  20,
  "a1", "00-00-2",  1.5,  55,
  "a1", "00-00-2",  2.5,  60
)
GT <- GeoTox() |>
  set_sample(sample_df) |>
  set_simulated_css(css_df) |>
  add_exposure_rate_params() |>
  add_hill_params(fit_hill(hill_df, assay = "assay", substance = "casn")) |>
  simulate_population(exposure = exposure_df) |>
  calc_internal_dose() |>
```

```
  calc_invitro_concentration()

# Calculate risk
GT <- GT |> calc_risk()

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at relevant table
dplyr::tbl(con, "risk") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`calc_sensitivity` *Calculate risk sensitivity to a single variable*

---

## Description

Compute risk by varying one variable while holding others fixed.

## Usage

```
calc_sensitivity(
  GT,
  vary = c("age", "weight", "css_params", "fit_params", "C_ext"),
  overwrite = FALSE,
  rate_extra_cols = NULL,
  expos_mean = NULL,
  expos_sd = NULL,
  max_mult = 1.5
)
```

## Arguments

| | |
|---|---|
| `GT` | GeoTox object. |
| `vary` | Variable to vary. One of "age", "weight", "css_params", "fit_params", or "C_ext". |
| `overwrite` | Logical indicating whether to overwrite existing sensitivity analysis results in the GeoTox database. |
| `rate_extra_cols` | Additional columns to match from the 'exposure_rate_params' table (default NULL). |
| `expos_mean` | Column name of exposure concentration mean in the 'exposure' table (default "mean"). |
| `expos_sd` | Column name of exposure concentration standard deviation in the 'exposure' table (default "sd"). |
| `max_mult` | Upper bound multiplier for max response (default 1.5). |

## Details

The sensitivity analysis makes use of the $C_{ss}$ values stored in the 'fixed_css' table of the GeoTox database. These values are determined using the pre-simulated $C_{ss}$ values supplied to `set_simulated_css()` and can be set using `set_fixed_css()` prior to running this function. This step is automatically done when using `simulate_population()` with `sample_css = TRUE`.

There are five options for the `vary` argument:

**age** $C_{ss}$ values from the 'age' column of the 'fixed_css' table are used. For other cases, exposure rates are re-simulated using the median age by location in the 'sample' table by calling `simulate_exposure_rate()` with `sensitivity = TRUE`.

**weight** $C_{ss}$ values from the 'weight' column of the 'fixed_css' table are used.

**css_params** $C_{ss}$ values from the 'params' column of the 'fixed_css' table are used.

**fit_params** $C_{ss}$ values from the 'other' column of the 'fixed_css' table are used. For other cases, the standard deviation of dose-response model fit parameters are set to zero by calling `calc_risk()` with `fixed = TRUE`.

**C_ext** $C_{ss}$ values from the 'other' column of the 'fixed_css' table are used. For other cases, external concentrations are re-simulated with standard deviations set to zero by calling `simulate_exposure()` with `sensitivity = TRUE`.

In all cases above, the resulting risk table is named 'risk_sensitivity_~vary~' (e.g., 'risk_sensitivity_age') in the GeoTox database.

Inputs `rate_extra_cols`, `expos_mean`, and `expos_sd` do not need to be specified again if they were already provided in a previous call and are set in the GeoTox parameters (`GT$par`).

## Value

The updated GeoTox object, invisibly.

## See Also

`set_simulated_css()`, `set_fixed_css()`, `sensitivity_analysis()`

## Examples

```
# Example setup is shown below in \dontrun().
# Pre-generated results will be loaded instead to avoid long example runtime.

## Not run:
# Setup required tables
sample_df <- tibble::tribble(
  ~FIPS, ~age, ~weight,
  10000, 25, "Normal",
  10000, 35,  "Obese",
  20000, 50, "Normal"
)
exposure_df <- tibble::tribble(
  ~FIPS, ~casn, ~route, ~mean, ~sd,
  10000, "00-00-1", "inhalation", 10, 1,
  10000, "00-00-2", "inhalation", 20, 1,
```

```
  20000, "00-00-1", "inhalation", 30, 1,
  20000, "00-00-2", "inhalation", 40, 1
)
css_df <- tibble::tribble(
  ~casn, ~age_lb, ~age_ub, ~weight, ~css,
  "00-00-1",  0, 49, "Normal", 21,
  "00-00-1", 50, 99, "Normal", 22,
  "00-00-1",  0, 49,  "Obese", 61,
  "00-00-1", 50, 99,  "Obese", 62,
  "00-00-2",  0, 49, "Normal", 11,
  "00-00-2", 50, 99, "Normal", 12,
  "00-00-2",  0, 49,  "Obese", 31,
  "00-00-2", 50, 99,  "Obese", 32
)
hill_df <- tibble::tribble(
  ~assay, ~model, ~casn, ~logc, ~resp,
  "a1", "human", "00-00-1",    0,  10,
  "a1", "human", "00-00-1",    1,  20,
  "a1", "human", "00-00-1",    2,  80,
  "a1", "human", "00-00-1",    3, 100,
  "a1", "human", "00-00-2", -0.5,   5,
  "a1", "human", "00-00-2",  0.5,  20,
  "a1", "human", "00-00-2",  1.5,  55,
  "a1", "human", "00-00-2",  2.5,  60,
  "a2",   "rat", "00-00-1",   -1,   0,
  "a2",   "rat", "00-00-1",    0,  10,
  "a2",   "rat", "00-00-1",    1,  30,
  "a2",   "rat", "00-00-1",    2,  40
)
set.seed(1234)
GT <- GeoTox() |>
  set_sample(sample_df) |>
  set_simulated_css(css_df) |>
  add_exposure_rate_params() |>
  add_hill_params(fit_hill(
    hill_df, assay = c(name = "assay", model = "model"), substance = "casn"
  )) |>
  simulate_population(exposure = exposure_df) |>
  calc_response()

# Calculate sensitivity to age
GT <- GT |> calc_sensitivity("age")

## End(Not run)

# Load results from pre-generated database for this example
temp_dir <- tempdir()
zip::unzip(
  system.file("extdata", "sensitivity.duckdb.zip", package = "GeoTox"),
  junkpaths = TRUE,
  exdir = temp_dir
)
GT <- GeoTox(paste0(temp_dir, "/sensitivity.duckdb"))
```

```
# Open a connection to GeoTox database
con <- get_con(GT)

# Look at relevant table

dplyr::tbl(con, "risk_sensitivity_age") |> dplyr::collect()

# Compared to baseline risk table
dplyr::tbl(con, "risk") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

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

---

## Description

Fit a 2-parameter (fixed slope) or 3-parameter (variable slope) Hill model to concentration-response data.

## Usage

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

## Arguments

| | |
|---|---|
| `x` | Data frame of dose response data. |
| `conc` | Column name of base-10 log scaled concentration (default "logc"). |
| `resp` | Column name of response (default "resp"). |
| `fixed_slope` | Logical indicating whether to fit a 2-parameter (TRUE) or 3-parameter (FALSE) Hill model (default TRUE). |
| `assay` | Column name of assay identifier(s) (optional, default NULL). |
| `substance` | Column name of substance identifier(s) (optional, default NULL). |

## Details

The input x data frame must contain columns specified by conc and resp arguments representing the log10-transformed concentration and response, respectively.

Optional assay and substance identifiers can be named vectors that are used to fit multiple substances and/or assays. For example, assay = c(name = "assay", "model") would indicate that x contains both and assay name and model. The name = "assay" part would rename the "assay" column in x to "name" in the 'assay' table when aded with add_hill_params().

Returned column '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

A list with elements 'fit', 'assay', 'substance'. The 'fit' element is a data frame of fit parameters, while the 'assay' and 'substance' elements indicate the column names used for assay and substance identifiers, respectively.

## See Also

add_hill_params()

## Examples

```
hill_df <- tibble::tribble(
  ~assay, ~model, ~casn, ~logc, ~resp,
  "a1", "human", "00-00-1",    0,  10,
  "a1", "human", "00-00-1",    1,  20,
  "a1", "human", "00-00-1",    2,  80,
  "a1", "human", "00-00-1",    3, 100,
  "a1", "human", "00-00-2", -0.5,   5,
  "a1", "human", "00-00-2",  0.5,  20,
  "a1", "human", "00-00-2",  1.5,  55,
  "a1", "human", "00-00-2",  2.5,  60,
  "a2",   "rat", "00-00-1",   -1,   0,
  "a2",   "rat", "00-00-1",    0,  10,
  "a2",   "rat", "00-00-1",    1,  30,
  "a2",   "rat", "00-00-1",    2,  40
)

# Fit 2-parameter Hill model
fit_hill(
  hill_df, assay = c(name = "assay", model = "model"), substance = "casn"
)

# Fit 3-parameter Hill model
fit_hill(hill_df, assay = "assay", substance = "casn", fixed_slope = FALSE)
```

GeoTox *GeoTox S3 object*

## Description

Create a GeoTox object and connect to the underlying database.

## Usage

```
GeoTox(dbname = tempfile(fileext = ".duckdb"), reset_seed = FALSE, ...)

get_con(GT)
```

## Arguments

| | |
|---|---|
| dbname | Database file name. Default is a temporary file. |
| reset_seed | Logical indicating whether to reset the user's global .Random.seed after certain database operations (default FALSE). |
| ... | Additional arguments passed to DBI::dbConnect(). |
| GT | GeoTox object. |

## Details

The dbname will point to a DuckDB database file. If the file does not already exist, a new database will be created. Additional arguments passed via ... will be forwarded to DBI::dbConnect().

The reset_seed parameter is necessary for replicating results from the previous GeoTox implementation. Some database functions create temporary tables where a random string is appended to the table name. This advances the .Random.seed state and causes functions that use randomness to produce different results between the previous and current implementations. Setting reset_seed = TRUE will reset the user's .Random.seed to the value it had before certain database operations.

Various parameters will be stored in the 'par' table. These parameters will also be loaded into the GeoTox object as a list in GT$par. The value for reset_seed can only be set on initial GeoTox object creation and will be loaded from the 'par' table for existing databases.

## Value

For GeoTox(), a GeoTox S3 object. For get_con(), a database connection.

## Examples

```
# Create a GeoTox object
GT <- GeoTox()

# Open a connection to GeoTox database
con <- get_con(GT)

# List database tables
```

```
DBI::dbListTables(con)

# Look at the GT parameters
dplyr::tbl(con, "par") |> dplyr::collect()
str(GT$par)

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`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** 2020 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 7/1/2020 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 2020.

---

`get_assay_table` *Get risk metric results*

---

## Description

Several functions used to fetch risk metric results from risk tables in a GeoTox database. The outputs of these functions are useful for plotting or further analysis.

## Usage

```
get_assay_table(GT)

get_risk_quantiles(
  GT,
  metric = c("GCA.Eff", "IA.Eff", "GCA.HQ.10", "IA.HQ.10"),
  quantiles = c(0.1, 0.25, 0.5, 0.75, 0.9),
  table_name = "risk"
)

get_risk_sensitivity(
  GT,
  metric = c("GCA.Eff", "IA.Eff", "GCA.HQ.10", "IA.HQ.10"),
  assay = NULL
)

get_risk_values(
  GT,
  metric = c("GCA.Eff", "IA.Eff", "GCA.HQ.10", "IA.HQ.10"),
  assay = NULL,
  table_name = "risk"
)
```

## Arguments

| | |
|---|---|
| GT | GeoTox object. |
| metric | Risk metric to retrieve. One of "GCA.Eff", "IA.Eff", "GCA.HQ.10", or "IA.HQ.10". |
| quantiles | Numeric vector of quantiles to calculate. |
| table_name | Name of the risk table to query (default "risk"). |
| assay | Named vector specifying an assay filter (default NULL). |

## Details

Normally an 'assay' table is created when adding the Hill model concentration-response fit parameters with `add_hill_params()`. If no `assay` input is specified in `fit_hill()`, then an 'assay' table will not be created. For `get_risk_values()`, the `assay` parameter can be used to filter results based on assay details stored in the 'assay' table. This is useful when multiple assays are available in the database. The `assay` input should be a named vector specifying the column name and value to filter by, e.g. `assay = c(endp = "mortality")`. For `get_risk_quantiles()`, if there is no assay data in the GeoTox database, then the output "assay_id" column will be filled with NA values. If there is assay data, use `get_assay_table()` to retrieve assay information and link the "assay_id" to assay details.

Use the `table_name` parameter to specify which risk table to query. There can be several risk tables in a GeoTox database. The default table is named "risk" and is created by either `calc_risk()` or the wrapper function `calc_response()`. Sensitivity analysis results created by either `calc_sensitivity()` or the wrapper function `sensitivity_analysis()` are stored in other tables with names like "risk_sensitivity_age", etc. Refer to `calc_sensitivity()` to see which tables may be available.

get_risk_sensitivity() is a wrapper function for get_risk_values() around all risk tables created by the original risk computation and subsequent sensitivity analysis. The column names are "baseline" for the original risk table, while the other column names correspond to the parameter varied in the sensitivity analysis.

## Value

A data frame or vector.

## See Also

calc_risk(), calc_response(), calc_sensitivity(), sensitivity_analysis(), fit_hill(), add_hill_params()

## Examples

```
# Example setup is shown below in \dontrun().
# Pre-generated results will be loaded instead to avoid long example runtime.

## Not run:
# Setup required tables
sample_df <- tibble::tribble(
  ~FIPS, ~age, ~weight,
  10000, 25, "Normal",
  10000, 35,  "Obese",
  20000, 50, "Normal"
)
exposure_df <- tibble::tribble(
  ~FIPS, ~casn, ~route, ~mean, ~sd,
  10000, "00-00-1", "inhalation", 10, 1,
  10000, "00-00-2", "inhalation", 20, 1,
  20000, "00-00-1", "inhalation", 30, 1,
  20000, "00-00-2", "inhalation", 40, 1
)
css_df <- tibble::tribble(
  ~casn, ~age_lb, ~age_ub, ~weight, ~css,
  "00-00-1",  0, 49, "Normal", 21,
  "00-00-1", 50, 99, "Normal", 22,
  "00-00-1",  0, 49,  "Obese", 61,
  "00-00-1", 50, 99,  "Obese", 62,
  "00-00-2",  0, 49, "Normal", 11,
  "00-00-2", 50, 99, "Normal", 12,
  "00-00-2",  0, 49,  "Obese", 31,
  "00-00-2", 50, 99,  "Obese", 32
)
hill_df <- tibble::tribble(
  ~assay, ~model, ~casn, ~logc, ~resp,
  "a1", "human", "00-00-1",    0,  10,
  "a1", "human", "00-00-1",    1,  20,
  "a1", "human", "00-00-1",    2,  80,
  "a1", "human", "00-00-1",    3, 100,
  "a1", "human", "00-00-2", -0.5,   5,
```

```
  "a1", "human", "00-00-2",  0.5,  20,
  "a1", "human", "00-00-2",  1.5,  55,
  "a1", "human", "00-00-2",  2.5,  60,
  "a2",   "rat", "00-00-1",   -1,   0,
  "a2",   "rat", "00-00-1",    0,  10,
  "a2",   "rat", "00-00-1",    1,  30,
  "a2",   "rat", "00-00-1",    2,  40
)
set.seed(1234)
GT <- GeoTox() |>
  set_sample(sample_df) |>
  set_simulated_css(css_df) |>
  add_exposure_rate_params() |>
  add_hill_params(fit_hill(
    hill_df, assay = c(name = "assay", model = "model"), substance = "casn"
  )) |>
  simulate_population(exposure = exposure_df) |>
  calc_response() |>
  sensitivity_analysis()

## End(Not run)

# Load results from pre-generated database for this example
temp_dir <- tempdir()
zip::unzip(
  system.file("extdata", "sensitivity.duckdb.zip", package = "GeoTox"),
  junkpaths = TRUE,
  exdir = temp_dir
)
GT <- GeoTox(paste0(temp_dir, "/sensitivity.duckdb"))

# Look at 'assay' table contents
get_assay_table(GT)

# Get "GCA.HQ.10" values from 'risk' table for the "a1" assay
get_risk_values(GT, metric = "GCA.HQ.10", assay = c(name = "a1"))

# Get "IA.Eff" values from all risk tables for the "a2" assay
get_risk_sensitivity(GT, metric = "IA.Eff", assay = c(name = "a2"))

# Get "GCA.Eff" quantiles from 'risk' table
get_risk_quantiles(GT, metric = "GCA.Eff", quantiles = c(0.25, 0.5, 0.75))

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at the 'risk' table contents
dplyr::tbl(con, "risk") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

`get_concentration_mean` *Get concentration mean values*

## Description

Calculates the concentration mean values for each substance and route at each location based on the data in the 'concentration' and 'sample' tables of the GeoTox database. The output of this function is useful for plotting or further analysis.

## Usage

```
get_concentration_mean(GT, col)
```

## Arguments

| | |
|---|---|
| `GT` | GeoTox object. |
| `col` | Column name in the 'concentration' table for which to calculate the mean, grouped by substance, route, and location. |

## Details

The `col` parameter specifies which column in the 'concentration' table to use for the mean calculation. The available choices will depend on what data has been stored in the 'concentration' table during the simulation process.

## Value

A data frame.

## Examples

```
# Setup required tables
exposure_df <- tibble::tribble(
  ~FIPS, ~casn, ~route, ~mean, ~sd,
  10000, "00-00-1", "inhalation", 10, 1,
  10000, "00-00-2", "inhalation", 20, 1,
  20000, "00-00-1", "inhalation", 30, 1,
  20000, "00-00-2", "inhalation", 40, 1
)
GT <- GeoTox() |>
  add_exposure(exposure_df) |>
  simulate_exposure(n = 100)

# Calculate mean external concentration by substance and location
get_concentration_mean(GT, "C_ext")

# Open a connection to GeoTox database
con <- get_con(GT)
```

```
# Look at column names in the 'concentration' table
dplyr::tbl(con, "concentration") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`sample_simulated_css` *Sample from pre-simulated steady-state plasma concentrations*

---

## Description

Sample steady-state plasma concentrations ($C_{ss}$) for individuals from pre-simulated values stored in the 'simulated_css' table in a GeoTox database.

## Usage

```
sample_simulated_css(GT, css_extra_cols = NULL, substance_order = NULL)
```

## Arguments

| | |
|---|---|
| `GT` | GeoTox object. |
| `css_extra_cols` | Additional columns to match from the 'simulated_css' table (default NULL). |
| `substance_order` | Named list specifying order of substance evaluation (default NULL). |

## Details

The $C_{ss}$ values are sampled from the 'simulated_css' table, which must already exist in the GeoTox database; it can be created using `set_simulated_css()`. The minimum characteristics that are used to match individuals to sets of $C_{ss}$ values are age and weight category ("Normal" or "Obese"). Additional columns (which must exist in both the 'simulated_css' and 'sample' tables) can be specified using the `css_extra_cols` argument. If `css_exta_cols` is provided it will be added to the GeoTox object parameter list, `GT$par`.

In addition to the 'simulated_css' table, both 'sample' and 'concentration' tables must also exist in the GeoTox database. The 'sample' table must contain the characteristics of individuals (age, weight category, and any additional columns specified in `css_extra_cols`). One way to create this 'sample' table is by using `set_sample()`. The 'concentration' table will typically be created using `simulate_exposure()` and will be populated with rows for each individual and substance combination where exposure data is available. The sampled $C_{ss}$ values will be stored in a new "C_ss" column in the 'concentration' table.

The `substance_order` argument can be used to specify the order in which substances are evaluated when sampling $C_{ss}$ values. The default is to evaluate substances in the order they appear in the 'substance' table. However, if a different order is needed for some reason (e.g., replication of results from a previous GeoTox implementation), the user can provide a named list where the name is the column in the 'substance' table to use for ordering (e.g., "casn") and the value is a vector of substance identifiers in the desired order. If provided, the `substance_order` will be added to the GeoTox object parameter list, `GT$par`.

## Value

The updated GeoTox object, invisibly.

## See Also

set_simulated_css(), simulate_population()

## Examples

```
# Create required tables
sample_df <- tibble::tribble(
  ~FIPS, ~age, ~weight,
  10000, 25, "Normal",
  10000, 35,  "Obese",
  20000, 50, "Normal"
)
exposure_df <- tibble::tribble(
  ~FIPS, ~casn, ~route, ~mean, ~sd,
  10000, "00-00-1", "inhalation", 10, 1,
  10000, "00-00-2", "inhalation", 20, 1,
  20000, "00-00-1", "inhalation", 30, 1,
  20000, "00-00-2", "inhalation", 40, 1
)
# Note: normally the css_df would have many more rows for each combination of
# the non-'css' columns to allow for sampling.
css_df <- tibble::tribble(
  ~casn, ~age_lb, ~age_ub, ~weight, ~css,
  "00-00-1",  0, 49, "Normal",  1,
  "00-00-1", 50, 99, "Normal",  2,
  "00-00-1",  0, 49,  "Obese", 11,
  "00-00-1", 50, 99,  "Obese", 12,
  "00-00-2",  0, 49, "Normal", 21,
  "00-00-2", 50, 99, "Normal", 22,
  "00-00-2",  0, 49,  "Obese", 31,
  "00-00-2", 50, 99,  "Obese", 32
)
GT <- GeoTox() |>
  set_sample(sample_df) |>
  add_exposure(exposure_df) |>
  simulate_exposure() |>
  set_simulated_css(css_df)

# Sample simulated C_ss values
GT <- GT |> sample_simulated_css()

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at created tables. sample_simulated_css() generated the 'C_ss' column
# of the 'concentration' table.

dplyr::tbl(con, "concentration") |> dplyr::collect()
```

```
dplyr::tbl(con, "sample") |> dplyr::collect()

dplyr::tbl(con, "location") |> dplyr::collect()

dplyr::tbl(con, "substance") |> dplyr::collect()

dplyr::tbl(con, "route") |> dplyr::collect()

# Replace sample and css tables with new data including an extra column
# Limit to a single substance for simplicity
sample_df <- tibble::tribble(
  ~FIPS, ~age, ~weight, ~sign,
  10000, 25, "Normal", "+",
  10000, 35,  "Obese", "-",
  20000, 50, "Normal", "+"
)
css_df <- tibble::tribble(
  ~casn, ~age_lb, ~age_ub, ~weight, ~css, ~sign,
  "00-00-1",  0, 49, "Normal",   1, "+",
  "00-00-1", 50, 99, "Normal",   2, "+",
  "00-00-1",  0, 49,  "Obese",  11, "+",
  "00-00-1", 50, 99,  "Obese",  12, "+",
  "00-00-1",  0, 49, "Normal",  -1, "-",
  "00-00-1", 50, 99, "Normal",  -2, "-",
  "00-00-1",  0, 49,  "Obese", -11, "-",
  "00-00-1", 50, 99,  "Obese", -12, "-"
)
GT <- GT |>
 set_sample(sample_df, overwrite = TRUE) |>
 simulate_exposure() |>
 set_simulated_css(css_df, overwrite = TRUE)

# Sample simulated C_ss values with extra column
# Notice how the extra column name is added to GT$par
str(GT$par)
GT <- GT |> sample_simulated_css(css_extra_cols = "sign")
str(GT$par)

# Look at new 'concentration' table. Values will be missing for substance 2
# since it is not in the new css_df.
dplyr::tbl(con, "concentration") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

sensitivity_analysis *Perform sensitivity analysis*

---

## Description

Calculate risk sensitivity to all available `vary` parameters in `calc_sensitivity()`.

## Usage

```
sensitivity_analysis(GT, max_mult = c(1.5, 1.5, 1.5, 1.5, 1.5), ...)
```

## Arguments

| | |
|---|---|
| `GT` | GeoTox object. |
| `max_mult` | Vector of length 5 containing upper bound multipliers for max response (default 1.5). |
| `...` | Additional arguments passed to each call of `calc_sensitivity()`. |

## Details

Sensitivity is calculated in the order: age, weight, css_params, fit_params, C_ext. The `max_mult` vector allows specifying different upper bound multipliers for each parameter.

## Value

The updated GeoTox object, invisibly.

## See Also

`calc_sensitivity()`

## Examples

```
# Example setup is shown below in \dontrun().
# Pre-generated results will be loaded instead to avoid long example runtime.

## Not run:
# Setup required tables
sample_df <- tibble::tribble(
  ~FIPS, ~age, ~weight,
  10000, 25, "Normal",
  10000, 35,  "Obese",
  20000, 50, "Normal"
)
exposure_df <- tibble::tribble(
  ~FIPS, ~casn, ~route, ~mean, ~sd,
  10000, "00-00-1", "inhalation", 10, 1,
  10000, "00-00-2", "inhalation", 20, 1,
  20000, "00-00-1", "inhalation", 30, 1,
  20000, "00-00-2", "inhalation", 40, 1
)
css_df <- tibble::tribble(
  ~casn, ~age_lb, ~age_ub, ~weight, ~css,
  "00-00-1",  0, 49, "Normal", 21,
```

```
  "00-00-1", 50, 99, "Normal", 22,
  "00-00-1",  0, 49,  "Obese", 61,
  "00-00-1", 50, 99,  "Obese", 62,
  "00-00-2",  0, 49, "Normal", 11,
  "00-00-2", 50, 99, "Normal", 12,
  "00-00-2",  0, 49,  "Obese", 31,
  "00-00-2", 50, 99,  "Obese", 32
)
hill_df <- tibble::tribble(
  ~assay, ~model, ~casn, ~logc, ~resp,
  "a1", "human", "00-00-1",    0,  10,
  "a1", "human", "00-00-1",    1,  20,
  "a1", "human", "00-00-1",    2,  80,
  "a1", "human", "00-00-1",    3, 100,
  "a1", "human", "00-00-2", -0.5,   5,
  "a1", "human", "00-00-2",  0.5,  20,
  "a1", "human", "00-00-2",  1.5,  55,
  "a1", "human", "00-00-2",  2.5,  60,
  "a2",   "rat", "00-00-1",   -1,   0,
  "a2",   "rat", "00-00-1",    0,  10,
  "a2",   "rat", "00-00-1",    1,  30,
  "a2",   "rat", "00-00-1",    2,  40
)
set.seed(1234)
GT <- GeoTox() |>
  set_sample(sample_df) |>
  set_simulated_css(css_df) |>
  add_exposure_rate_params() |>
  add_hill_params(fit_hill(
    hill_df, assay = c(name = "assay", model = "model"), substance = "casn"
  )) |>
  simulate_population(exposure = exposure_df) |>
  calc_response()

# Perform sensitivity analysis
GT <- GT |> sensitivity_analysis()

## End(Not run)

# Load results from pre-generated database for this example
temp_dir <- tempdir()
zip::unzip(
  system.file("extdata", "sensitivity.duckdb.zip", package = "GeoTox"),
  junkpaths = TRUE,
  exdir = temp_dir
)
GT <- GeoTox(paste0(temp_dir, "/sensitivity.duckdb"))

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at relevant table
```

```
dplyr::tbl(con, "risk_sensitivity_age") |> dplyr::collect()

dplyr::tbl(con, "risk_sensitivity_weight") |> dplyr::collect()

dplyr::tbl(con, "risk_sensitivity_css_params") |> dplyr::collect()

dplyr::tbl(con, "risk_sensitivity_fit_params") |> dplyr::collect()

dplyr::tbl(con, "risk_sensitivity_C_ext") |> dplyr::collect()

# Compared to baseline risk table
dplyr::tbl(con, "risk") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`set_boundary` *Store and retrieve boundary geometries*

---

## Description

Boundary information is stored as serialized 'BLOB' objects in the 'boundary' table.

## Usage

```
set_boundary(GT, df_list, location = "county", overwrite = FALSE)

get_boundary(GT)
```

## Arguments

| | |
|---|---|
| `GT` | GeoTox object. |
| `df_list` | Named list of data frames containing boundary geometries as sf objects. |
| `location` | Name of element in `df_list` that contains location boundary information (default "county"). |
| `overwrite` | Logical indicating whether to overwrite existing 'boundary' table. |

## Details

NOTE: This function requires the `sf` package to be installed.

This function takes a named list of sf objects and stores them in the database. If a location boundary (default "county") is provided, then the non-geometry fields will be added to the 'location' table and replaced with a 'location_id' value so they can be linked to data in the 'sample' table.

## Value

For `set_boundary()`, the same GeoTox object, invisibly. For `get_boundary()`, a data frame with columns 'id' (boundary name) and 'data' (sf object).

## Examples

```
# Setup sf objects
county <- sf::st_sf(
  FIPS = c(10000, 20000),
  geometry = sf::st_sfc(sf::st_point(1:2), sf::st_point(3:4))
)
state <- sf::st_sf(
  STATE = "XYZ",
  geometry = sf::st_sfc(sf::st_point(5:6))
)
df_list <- list(county = county, state = state)

# Add boundary to GeoTox database
GT <- GeoTox() |> set_boundary(df_list)

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at created tables

dplyr::tbl(con, "boundary") |> dplyr::collect()

dplyr::tbl(con, "location") |> dplyr::collect()

# Retrieve boundary from GeoTox database
boundary <- get_boundary(GT)
boundary
boundary |> tibble::deframe()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`set_fixed_css` *Prepare steady-state plasma concentrations for sensitivity analysis*

---

## Description

Create the 'fixed_css' table in the GeoTox database, which contains values of steady-state plasma concentrations ($C_{ss}$) for sensitivity analysis.

## Usage

```
set_fixed_css(GT, substance_order = NULL)
```

## Arguments

| | |
|---|---|
| `GT` | GeoTox object. |
| `substance_order` | Named list specifying order of substance evaluation (default NULL). |

## Details

Several tables are required in the GeoTox database before the 'fixed_css' table can be created. Typically, `set_fixed_css()` is called after using `sample_simulated_css()`, at which point all required tables will be present.

The resulting 'fixed_css' table is used for sensitivity analysis where one parameter is allowed to vary at a time while all other parameters are held constant at fixed values.

The `substance_order` argument can be used to specify the order in which substances are evaluated when sampling $C_{ss}$ values. The default is to evaluate substances in the order they appear in the 'substance' table. However, if a different order is needed for some reason (e.g., replication of results from a previous GeoTox implementation), the user can provide a named list where the name is the column in the 'substance' table to use for ordering (e.g., "casn") and the value is a vector of substance identifiers in the desired order. If provided, the `substance_order` will be added to the GeoTox object parameter list, `GT$par`. This is only applicable to the "params" section described below.

**Table 'fixed_css' columns::**

**age** Median pre-simulated $C_{ss}$ values are computed by age group for each substance, then the median $C_{ss}$ values are assigned to each individual based on their age group.

**weight** Median pre-simulated $C_{ss}$ values are computed by weight category for each substance, then the median $C_{ss}$ values are assigned to each individual based on their weight category.

**params** Individuals are assigned the median age of their location and pre-simulated $C_{ss}$ values for the "Normal" weight category are sampled.

**other** Median sampled $C_{ss}$ values are computed across all substances for each location after mean-imputation of missing $C_{ss}$ values for each substance and location. The median $C_{ss}$ values are assigned to each individual based on their location.

## Value

The updated GeoTox object, invisibly.

## See Also

`sample_simulated_css()`, `simulate_population()`

## Examples

```
# Create required tables
sample_df <- tibble::tribble(
  ~FIPS, ~age, ~weight,
  10000, 25, "Normal",
  10000, 35,  "Obese",
  20000, 50, "Normal"
)
exposure_df <- tibble::tribble(
  ~FIPS, ~casn, ~route, ~mean, ~sd,
  10000, "00-00-1", "inhalation", 10, 1,
  10000, "00-00-2", "inhalation", 20, 1,
  20000, "00-00-1", "inhalation", 30, 1,
  20000, "00-00-2", "inhalation", 40, 1
```

```
)
# Note: normally the css_df would have many more rows for each combination of
# the non-'css' columns to allow for sampling.
css_df <- tibble::tribble(
  ~casn, ~age_lb, ~age_ub, ~weight, ~css,
  "00-00-1",  0, 49, "Normal",  1,
  "00-00-1", 50, 99, "Normal",  2,
  "00-00-1",  0, 49,  "Obese", 11,
  "00-00-1", 50, 99,  "Obese", 12,
  "00-00-2",  0, 49, "Normal", 21,
  "00-00-2", 50, 99, "Normal", 22,
  "00-00-2",  0, 49,  "Obese", 31,
  "00-00-2", 50, 99,  "Obese", 32
)
GT <- GeoTox() |>
  set_sample(sample_df) |>
  add_exposure(exposure_df) |>
  simulate_exposure() |>
  set_simulated_css(css_df) |>
  sample_simulated_css()

# Set fixed C_ss values
GT <- GT |> set_fixed_css()

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at created tables
# Note: the 'age', 'weight', 'params', and 'other' columns of the
# 'fixed_css' table contain the C_ss values for sensitivity analysis.
# For example, the 'age' column doesn't contain ages, but C_ss values.

dplyr::tbl(con, "fixed_css") |> dplyr::collect()

dplyr::tbl(con, "concentration") |> dplyr::collect()

dplyr::tbl(con, "sample") |> dplyr::collect()

dplyr::tbl(con, "location") |> dplyr::collect()

dplyr::tbl(con, "substance") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`set_sample` *Set sample data*

---

## Description

Create the 'sample' table in the GeoTox database.

## Usage

```
set_sample(GT, df, location = "FIPS", overwrite = FALSE)
```

## Arguments

| | |
|---|---|
| GT | GeoTox object. |
| df | Data frame containing sample data. |
| location | Column name(s) in df that contain location identifier(s) (default "FIPS"). |
| overwrite | Logical indicating whether to overwrite existing 'sample' table and remove existing 'concentration' and 'risk' tables (default FALSE). |

## Details

The 'sample' table consists of individual characteristics and location information. At a minimum, it must contain columns with information on age, weight category, and location identifier(s).

When the df input contains information on age or weight category, the column names must be "age" and "weight", respectively. The location input (default "FIPS") can be a named vector to specify multiple identifier columns in df. For example, location = c(FIPS = "FIPS", state = "ST") would indicate that df contains both FIPS codes and state identifiers for locations. The location column(s) will be added to the 'location' table and replaced with a corresponding "location_id" column in the 'sample' table. The state = "ST" part in the example above would rename the "ST" column in df to "state" in the 'location' table.

There are several other functions that can be used to create or add to the 'sample' table: simulate_age() to generate age data, simulate_obesity() to generate weight category data, and simulate_population(), which is a wrapper function that can generate both age and weight category data along with other fields. set_sample() can be used in combination with these functions to first set some known sample data, e.g. age, and then simulate any missing fields, e.g. weight category.

If overwrite = TRUE, any existing 'concentration' and 'risk' tables will be dropped before creating the 'sample' table. This is because the existing concentration and risk data would no longer be valid for the new sample data. Further downstream tables are not dropped automatically, but can be updated during subsequent simulation and analysis steps.

## Value

The same GeoTox object, invisibly.

## See Also

simulate_age(), simulate_obesity(), simulate_population()

## Examples

```
# Example sample data
sample_df <- tibble::tribble(
  ~FIPS, ~age, ~weight,
  10000, 25, "Normal",
  10000, 35,  "Obese",
  20000, 50, "Normal"
```

```
)

# Set sample data
GT <- GeoTox() |> set_sample(sample_df)

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at created tables

dplyr::tbl(con, "sample") |> dplyr::collect()

dplyr::tbl(con, "location") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`set_simulated_css` *Set pre-simulated steady-state plasma concentrations*

---

## Description

Create the 'simulated_css' table in a GeoTox database, which contains pre-simulated steady-state plasma concentrations ($C_{ss}$).

## Usage

```
set_simulated_css(GT, df, substance = "casn", overwrite = FALSE)
```

## Arguments

| | |
|---|---|
| `GT` | GeoTox object. |
| `df` | Data frame containing simulated $C_{ss}$ values for groups of population characteristics. |
| `substance` | Column name(s) in `df` that contain substance identifier(s) (default "casn"). |
| `overwrite` | Logical indicating whether to overwrite existing 'simulated_css' table (default FALSE). |

## Details

The 'simulated_css' table is used by `sample_simulated_css()` to assign $C_{ss}$ values to individuals. The minimum required columns in the `df` data frame are "age_lb", "age_ub", "weight", "css", and at least one column with substance information (default "casn").

The values for "age_lb" and "age_ub" should be non-overlapping integers representing age ranges (in years). For example, two subsequent age groups might be `c(0, 4)` and `c(5, 9)`. The "weight" column should contain the weight category and contain values of either "Normal" or "Obese". The "css" column should contain the pre-simulated $C_{ss}$ values.

The substance input can be a named vector to specify multiple substance identifier columns in df. For example, c(casn = "casn", name = "chnm") would indicate that df contains both CAS numbers and chemical names for substances. The name = "chnm" part would rename the "chnm" column in df to "name" in the 'substance' table.

## Value

The same GeoTox object, invisibly.

## See Also

sample_simulated_css()

## Examples

```
# Example pre-simulated C_ss data
# Note: normally the css_df would have many more rows for each combination of
# the non-'css' columns to allow for sampling.
css_df <- tibble::tribble(
  ~casn, ~age_lb, ~age_ub, ~weight, ~css,
  "00-00-1",  0, 49, "Normal",  1,
  "00-00-1", 50, 99, "Normal",  2,
  "00-00-1",  0, 49,  "Obese", 11,
  "00-00-1", 50, 99,  "Obese", 12,
  "00-00-2",  0, 49, "Normal", 21,
  "00-00-2", 50, 99, "Normal", 22,
  "00-00-2",  0, 49,  "Obese", 31,
  "00-00-2", 50, 99,  "Obese", 32
)

# Set simulated C_ss values
GT <- GeoTox() |> set_simulated_css(css_df)

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at relevant tables

dplyr::tbl(con, "simulated_css") |> dplyr::collect()

dplyr::tbl(con, "substance") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

# simulate_age *Simulate age values*

---

## Description

Simulate 'age' values to be stored in the 'sample' table of a GeoTox database.

## Usage

```
simulate_age(GT, n = 1000, overwrite = FALSE)
```

## Arguments

| | |
|---|---|
| GT | GeoTox object. |
| n | Number of individuals to simulate per location (default 1000). Ignored if 'sample' table already exists. |
| overwrite | Logical indicating whether to overwrite existing 'age' values in the 'sample' table (default FALSE). |

## Details

An 'age' table containing simulation data must already exist in the GeoTox database, which is added using add_age().

## Value

The same GeoTox object, invisibly.

## See Also

add_age(), simulate_population()

## Examples

```
# Example age simulation data
age_df <- data.frame(
  FIPS = rep(c(10000, 20000), each = 19),
  AGEGRP = rep(0:18, times = 2),
  TOT_POP = 0
)
# FIPS 10000, populate age group 40-44
age_df$TOT_POP[c(1, 10)] = 100
# FIPS 20000, populate age groups 50-59
age_df$TOT_POP[c(1, 12, 13) + 19] = c(200, 100, 100)

# Simulate age values
GT <- GeoTox() |>
  add_age(age_df) |>
  simulate_age(n = 5)

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at created tables
```

```
dplyr::tbl(con, "sample") |> dplyr::collect()

dplyr::tbl(con, "location") |> dplyr::collect()

# Overwrite existing age values
GT <- GT |> simulate_age(overwrite = TRUE)

# Look at updated 'sample' table
dplyr::tbl(con, "sample") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`simulate_exposure` *Simulate exposure concentrations*

---

## Description

Simulate external exposure (C_ext) values to be stored in the 'concentration' table of a GeoTox database.

## Usage

```
simulate_exposure(
  GT,
  n = 1000,
  overwrite = FALSE,
  expos_mean = NULL,
  expos_sd = NULL,
  sensitivity = FALSE
)
```

## Arguments

| | |
|---|---|
| `GT` | GeoTox object. |
| `n` | Number of individuals to simulate per location (default 1000). Ignored if 'sample' table already exists. |
| `overwrite` | Logical indicating whether to overwrite existing 'C_ext' values in the 'concentration' table (default FALSE). |
| `expos_mean` | Column name of exposure concentration mean in the 'exposure' table (default "mean"). |
| `expos_sd` | Column name of exposure concentration standard deviation in the 'exposure' table (default "sd"). |
| `sensitivity` | Logical indicating whether to simulate exposures for sensitivity analysis (default FALSE). |

## Details

An 'external' table containing simulation data must already exist in the GeoTox database, which is added using `add_exposure()`.

The inputs `expos_mean` and `expos_sd` will be assigned default values of "mean" and "sd", respectively, if not provided and not already set in the GeoTox object's parameters, `GT$par`. If not `NULL`, the provided values will also be saved to `GT$par`.

If `sensitivity = TRUE`, exposure concentrations will be simulated for sensitivity analysis. Typically this shouldn't be used directly by the user, but rather called by `calc_sensitivity()`. In this case, the function will use the 'concentration_sensitivity' table instead of the 'concentration' table, and will assume that the 'sample' table already exists.

## Value

The updated GeoTox object, invisibly.

## See Also

`add_exposure()`, `simulate_population()`

## Examples

```
# Example exposure simulation data
exposure_df <- tibble::tribble(
  ~FIPS, ~casn, ~route, ~mean, ~sd,
  10000, "00-00-1", "inhalation", 10, 1,
  10000, "00-00-2", "inhalation", 20, 1,
  20000, "00-00-1", "inhalation", 30, 1,
  20000, "00-00-2", "inhalation", 40, 1
)

# Simulate C_ext values
GT <- GeoTox() |>
  add_exposure(exposure_df) |>
  simulate_exposure(n = 3)

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at created tables

dplyr::tbl(con, "concentration") |> dplyr::collect()

dplyr::tbl(con, "sample") |> dplyr::collect()

dplyr::tbl(con, "location") |> dplyr::collect()

dplyr::tbl(con, "substance") |> dplyr::collect()

dplyr::tbl(con, "route") |> dplyr::collect()
```

```
# Replace 'exposure' table with different column names
names(exposure_df)[4:5] <- c("mu", "sigma")
DBI::dbRemoveTable(con, "exposure")
GT |> add_exposure(exposure_df)

# Overwrite 'C_ext' values in existing 'concentration' table
# Must specify new 'exposure' column names
# Notice how the column names are added to GT$par
str(GT$par)
GT <- GT |>
  simulate_exposure(expos_mean = "mu", expos_sd = "sigma", overwrite = TRUE)
str(GT$par)

# Look at updated 'concentration' table
dplyr::tbl(con, "concentration") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`simulate_exposure_rate` *Simulate exposure rates*

---

## Description

Simulate exposure rate values for each sample and exposure route based on parameters in the `exposure_rate_params` table.

## Usage

```
simulate_exposure_rate(
  GT,
  rate_extra_cols = NULL,
  overwrite = FALSE,
  sensitivity = FALSE
)
```

## Arguments

| | |
|---|---|
| `GT` | GeoTox object. |
| `rate_extra_cols` | Additional columns to match from the 'exposure_rate_params' table (default NULL). |
| `overwrite` | Logical indicating whether to overwrite existing 'exposure_rate' table (default FALSE). |
| `sensitivity` | Logical indicating whether to simulate exposure rates for sensitivity analysis (default FALSE). |

## Details

An 'exposure_rate_params' table must already exist in the GeoTox database, which is added using `add_exposure_rate_params()`. There must also be a 'sample' table with an 'age' column, which can be created using `simulate_age()` or `set_sample()`. 'location' and 'route' tables must also exist, but they are created when adding the rate parameters and samples.

If `rate_exta_cols` is provided it will be added to the GeoTox object parameter list, `GT$par`. These columns must exist in the 'exposure_rate_params' table and will be used to match between the 'sample' and 'exposure_rate_params' tables when simulating exposure rates.

Typically this function will be called with `sensitivity` set to `FALSE`. In this case, the function will create an 'exposure_rate' table with simulated exposure rates for each sample and exposure route.

If `sensitivity` is `TRUE`, exposure rates will be simulated for sensitivity analysis. Typically this shouldn't be used directly by the user, but rather called by `calc_sensitivity()`. In this case, the existing 'exposure_rate' table will be copied to the 'exposure_rate_sensitivity' table where the rate values will be overwritten.

## Value

The updated GeoTox object, invisibly.

## See Also

`add_exposure_rate_params()`, `simulate_population()`

## Examples

```
# Setup required tables
# Note: 'gender' is ignored when using the default rate params
sample_df <- tibble::tribble(
  ~FIPS, ~age, ~weight, ~gender,
  10000, 25, "Normal",   "male",
  10000, 35,  "Obese",   "male",
  20000, 50, "Normal", "female"
)
GT <- GeoTox() |>
  add_exposure_rate_params() |>
  set_sample(sample_df)

# Simulate exposure rates
GT |> simulate_exposure_rate()

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at created tables

dplyr::tbl(con, "exposure_rate") |> dplyr::collect()

dplyr::tbl(con, "sample") |> dplyr::collect()

dplyr::tbl(con, "location") |> dplyr::collect()
```

```
dplyr::tbl(con, "route") |> dplyr::collect()

# Replace exposure rate params with a new table that includes gender
params_df <- tibble::tribble(
  ~age_lb, ~age_ub, ~gender, ~mean, ~sd,
   0, 49,   "male", 10, 1,
  50, 99,   "male", 20, 1,
   0, 49, "female", 30, 1,
  50, 99, "female", 40, 1
)
DBI::dbRemoveTable(con, "exposure_rate_params")
GT |> add_exposure_rate_params(params = params_df)

# Overwrite 'rate' values in existing 'exposure_rate' table
# Must specify additional column names
# Notice how the column names are added to GT$par
str(GT$par)
GT <- GT |>
  simulate_exposure_rate(rate_extra_cols = c("gender"), overwrite = TRUE)
str(GT$par)

# Look at updated 'exposure_rate' table
dplyr::tbl(con, "exposure_rate") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`simulate_obesity` *Simulate obesity values*

---

## Description

Simulate 'weight' category values to be stored in the 'sample' table of a GeoTox database.

## Usage

```
simulate_obesity(
  GT,
  n = 1000,
  overwrite = FALSE,
  obes_prev = NULL,
  obes_sd = NULL
)
```

## Arguments

| | |
|---|---|
| GT | GeoTox object. |

| | |
|---|---|
| n | Number of individuals to simulate per location (default 1000). Ignored if 'sample' table already exists. |
| overwrite | Logical indicating whether to overwrite existing 'weight' values in the 'sample' table (default FALSE). |
| obes_prev | Column name of obesity prevalence in the 'obesity' table (default "OBESITY_CrudePrev"). |
| obes_sd | Column name of obesity standard deviation in the 'obesity' table (default "OBESITY_SD"). |

## Details

An 'obesity' table containing simulation data must already exist in the GeoTox database, which is added using `add_obesity()`.

The inputs `obes_prev` and `obes_sd` will be assigned default values of "OBESITY_CrudePrev" and "OBESITY_SD", respectively, if not provided and not already set in the GeoTox object's parameters, `GT$par`. If not `NULL`, the provided values will also be saved to `GT$par`.

## Value

The updated GeoTox object, invisibly.

## See Also

`add_obesity()`, `simulate_population()`

## Examples

```
# Example obesity simulation data
obesity_df <- data.frame(
  FIPS = c(10000, 20000),
  OBESITY_CrudePrev = c(20, 80),
  OBESITY_SD = 5
)

# Simulate weight category values
GT <- GeoTox() |>
  add_obesity(obesity_df) |>
  simulate_obesity(n = 5)

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at created tables

dplyr::tbl(con, "sample") |> dplyr::collect()

dplyr::tbl(con, "location") |> dplyr::collect()

# Replace 'obesity' table with different column names
names(obesity_df)[2:3] <- c("prev", "sd")
DBI::dbRemoveTable(con, "obesity")
```

```
GT |> add_obesity(obesity_df)

# Overwrite 'weight' category values in existing 'sample' table
# Must specify new 'obesity' column names
# Notice how the column names are added to GT$par
str(GT$par)
GT <- GT |>
  simulate_obesity(obes_prev = "prev", obes_sd = "sd", overwrite = TRUE)
str(GT$par)

# Look at updated 'sample' table
dplyr::tbl(con, "sample") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

---

`simulate_population` *Simulate population characteristics and exposures*

---

## Description

Simulate age, weight category, exposure rates, and external exposures. Sample from pre-simulated steady-state plasma concentrations.

## Usage

```
simulate_population(
  GT,
  age = NULL,
  obesity = NULL,
  exposure = NULL,
  simulate_rate = TRUE,
  sample_css = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `GT` | GeoTox object. |
| `age` | Data frame with age data. |
| `obesity` | Data frame with obesity data. |
| `exposure` | Data frame with exposure data. |
| `simulate_rate` | Logical indicating whether to simulate exposure rates. This requires that exposure rate parameters have been added using `add_exposure_rate_params()`. |

`sample_css` Logical indicating whether to sample steady-state plasma concentrations ($C_{ss}$). This requires that a table of simulated $C_{ss}$ values has been set using `set_simulated_css()`. In addition, `set_fixed_css()` will be called after sampling to prepare $C_{ss}$ values for sensitivity analysis.

`...` Additional arguments passed to wrapped functions (see 'Additional arguments' section of 'Details').

## Details

This is a wrapper around several other functions:

- `add_age()` and `simulate_age()` for age simulation.
- `add_obesity()` and `simulate_obesity()` for weight category simulation.
- `add_exposure()` and `simulate_exposure()` for external exposure concentration simulation ($C_{ext}$).
- `simulate_exposure_rate()` for exposure rate simulation.
- `sample_simulated_css()` for sampling steady-state plasma concentrations ($C_{ss}$).
- `set_fixed_css()` to prepare $C_{ss}$ values for sensitivity analysis.

The user can provide data frames for age, obesity, and exposure data; if any of these are provided, the corresponding add and simulate functions will be called. The user can also specify whether to simulate exposure rates and sample $C_{ss}$ values using the `simulate_rate` and `sample_css` arguments, respectively. If `simulate_rate` is `TRUE`, exposure rate parameters must have been added using `add_exposure_rate_params()`. If `sample_css` is `TRUE`, a table of simulated $C_{ss}$ values must already exist using `set_simulated_css()`.

**Additional arguments::**

**n** Number of samples to simulate (default 1000). Used in `simulate_age()`, `simulate_obesity()`, and `simulate_exposure()`. Ignored if the 'sample' table already exists, in which case the existing sample sizes are used.

**location** Column name for location ID (default "FIPS"). Used in `add_age()`, `add_obesity()`, and `add_exposure()`.

**overwrite** Logical indicating whether to overwrite existing values (default FALSE). Used in `simulate_age()`, `simulate_obesity()`, `simulate_exposure_rate()`, and `simulate_exposure()`.

**substance** Column name for substance ID (default "casn"). Used in `add_exposure()`.

**route** Column name for exposure route (default "route"). Used in `add_exposure()`.

**rate_extra_cols** Additional columns to include in exposure_rate table. Used in `simulate_exposure_rate()`.

**obes_prev, obes_sd** Column names for obesity prevalence and standard deviation (default "OBESITY_CrudePrev" and "OBESITY_SD", respectively). Used in `simulate_obesity()`.

**expos_mean, expos_sd** Column names for exposure concentration mean and standard deviation (default "mean" and "sd", respectively). Used in `simulate_exposure()`.

**css_extra_cols** Additional columns to include in simulated_css table. Used in `sample_simulated_css()`.

**substance_order** Named list specifying order of substances. Used in `sample_simulated_css()` and `set_fixed_css()`.

## Value

The updated GeoTox object, invisibly.

## See Also

add_age(), simulate_age(), add_obesity(), simulate_obesity(), add_exposure(), simulate_exposure(), simulate_exposure_rate(), sample_simulated_css(), set_fixed_css()

## Examples

```
# Example simulation data

age_df <- data.frame(
  FIPS = rep(c(10000, 20000), each = 19),
  AGEGRP = rep(0:18, times = 2),
  TOT_POP = 0
)
# FIPS 10000, populate age group 40-44
age_df$TOT_POP[c(1, 10)] = 100
# FIPS 20000, populate age groups 50-59
age_df$TOT_POP[c(1, 12, 13) + 19] = c(200, 100, 100)

obesity_df <- data.frame(
  FIPS = c(10000, 20000),
  OBESITY_CrudePrev = c(20, 80),
  OBESITY_SD = 5
)

exposure_df <- tibble::tribble(
  ~FIPS, ~casn, ~route, ~mean, ~sd,
  10000, "00-00-1", "inhalation", 10, 1,
  10000, "00-00-2", "inhalation", 20, 1,
  20000, "00-00-1", "inhalation", 30, 1,
  20000, "00-00-2", "inhalation", 40, 1
)

# Note: normally the css_df would have many more rows for each combination of
# the non-'css' columns to allow for sampling.
css_df <- tibble::tribble(
  ~casn, ~age_lb, ~age_ub, ~weight, ~css,
  "00-00-1",  0, 49, "Normal",  1,
  "00-00-1", 50, 99, "Normal",  2,
  "00-00-1",  0, 49,  "Obese", 11,
  "00-00-1", 50, 99,  "Obese", 12,
  "00-00-2",  0, 49, "Normal", 21,
  "00-00-2", 50, 99, "Normal", 22,
  "00-00-2",  0, 49,  "Obese", 31,
  "00-00-2", 50, 99,  "Obese", 32
)

# Simulate population
GT <- GeoTox() |>
```

```
  add_exposure_rate_params() |>
  set_simulated_css(css_df) |>
  simulate_population(
    age = age_df,
    obesity = obesity_df,
    exposure = exposure_df,
    n = 3
  )

# Open a connection to GeoTox database
con <- get_con(GT)

# Look at created tables

dplyr::tbl(con, "concentration") |> dplyr::collect()

dplyr::tbl(con, "sample") |> dplyr::collect()

dplyr::tbl(con, "location") |> dplyr::collect()

dplyr::tbl(con, "substance") |> dplyr::collect()

dplyr::tbl(con, "route") |> dplyr::collect()

# Note: the 'age', 'weight', 'params', and 'other' columns of the
# 'fixed_css' table contain the C_ss values for sensitivity analysis.
# For example, the 'age' column doesn't contain ages, but C_ss values.
dplyr::tbl(con, "fixed_css") |> dplyr::collect()

# Clean up example
DBI::dbDisconnect(con)
file.remove(GT$db_info$dbdir)
```

# Index