Package: ValidationExplorer (via r-universe)

June 8, 2026

Title Simulation-Based Tools for Bioacoustic Study Design

Version 0.1.1

Description Many bioacoustic data workflows rely on manual review (i.e., validation) of a subset of call files to provide information to statistical models that account for misclassification by automated algorithms. Because manual review can be prohibitively expensive, simulation can be a valuable tool to aid the design of studies that use validation. This package provides user-friendly functions to reduce the programming burden of simulation studies that compare validation sampling designs. Simulations assume the count-detection model, which is a realistic model for bioacoustic data, especially for bats. For more information, see Oram et al. (2025) <doi:10.1214/25-AOAS2096>.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends nimble, R (>= 4.1.0)

Imports coda, dplyr, ggplot2, magrittr, purrr, rlang, rstan, stringr, tibble, tidyr, viridis

LazyData true

Suggests testthat (>= 3.0.0), withr

Config/testthat/edition 3

Config/Needs/website rmarkdown

URL https://j-oram.github.io/ValidationExplorer/

BugReports https://github.com/j-oram/ValidationExplorer/issues

Config/pak/sysreqs libglpk-dev make libicu-dev libxml2-dev

Repository https://j-oram.r-universe.dev

Date/Publication 2026-05-29 22:27:07 UTC

RemoteUrl https://github.com/j-oram/validationexplorer

RemoteRef HEAD

RemoteSha 71366175f2f22c6473218fda82084766ecfdf9b3

Contents

`example_output` *example_output*

Description

A dataframe in the format output from running run_sims. This exists solely for the purpose of making documentation of visualization functions easier.

- parameter. Parameters (lambda, psi, and theta for a two species assemblage).
- Mean. The posterior mean.
- SD. The standard deviation of the draws of reach parameter.
- Naive SE. The standard error, not accounting for the correlation in draws
- Time-series SE. The standard error, accounting for correlation in draws.
- quantiles. 2.5%, 25%, 50%, 75% and 97.5% quantiles.
- Rhat. The Gelman-Rubin statistic for each parameter.
- ess_bulk. The effective sample size in the bulk of the distribution.
- ess_tail. The effective sample size in the tails of the distribution.
- truth. The known true data generating value.
- capture. Did the 95% posterior interval contain the true value?

- converge. Was the Gelman-Rubin statistic near 1?
- theta_scenario. The ID for the classifier scenario.
- scenario. The index of the validation scenario.
- dataset. The dataset index.

Format

A dataframe containing the columns described below.

Source

A complete small-scale simulation study was run using the functions in `ValidationExplorer`.

Examples

```
head(example_output)

visualize_parameter_group(
  example_output,
  pars = "lambda",
  theta_scenario = "1",
  scenarios = 1:2
)
```

`example_val_sum` *example_val_sum: Example summaries of validated data*

Description

Example output from summarize_n_validated.

Format

A dataframe with 2 rows and 4 columns

Details

- theta_scenario. The ID of the classifier scenario.
- scenario. The validation scenario index.
- n_validated. The expected number of validated calls in an average dataset simulated under each scenario.

Source

Data was simulated using simulate_validatedData and summarized using summarize_n_validated.

Examples

```
example_val_sum
```

`mask_by_spp` *mask_by_spp: simulate a validation design*

Description

mask_by_spp: simulate a validation design

Usage

```
mask_by_spp(data, props_to_val)
```

Arguments

`data`	A dataframe containing the columns `site`, `visit`, `true_spp`, `id_spp`, `count`
`props_to_val`	a vector containing the proportion of recordings to validate for each species

Value

A list containing two elements: `final_df` and `data_sum`. `final_df` is a copy of the input `data` masked according to the validation design supplied by `props_to_val`. The second output, `data_sum` is a dataframe containing a summary of the number and proportion of ambiguous (i.e., not validated) recordings. It provides a check that the masking function is working correctly.

Examples

```
library(dplyr)

dat <- sim_dat()$full_df

head(dat)

dat <- dat %>% tidyr::uncount(weights = count, .remove = FALSE)
val_dat <- mask_by_spp(dat, props_to_val = c(rep(.1, 4), rep(.4, 4)))

val_dat$final_df %>% group_by(id_spp) %>%
  summarize(prop_vald = sum(!is.na(true_spp))/n())
```

`mask_FE_all_visits` *Mask a proportion of all visits: A function for simulating a fixed effort validation design.*

Description

Mask a proportion of all visits: A function for simulating a fixed effort validation design.

Usage

```
mask_FE_all_visits(df, effort_prop, seed = NULL)
```

Arguments

`df`	A dataframe object in the format of `full_dfs` output from simulate_validatedData
`effort_prop`	The proportion of recordings to be randomly sampled from the first visit for validation
`seed`	An optional random seed to make masking reproducible

Value

A dataframe object that is a copy of the input `df`, but with the appropriate level of effort according to a fixed effort validation design.

Examples

```
library(dplyr)

cd_data <- sim_dat()$full_df %>% tidyr::uncount(weights = count, .remove = FALSE)
FE_data <- mask_FE_all_visits(cd_data, effort_prop = 0.1)

head(cd_data)
head(FE_data)

FE_data %>%
  group_by(site, visit) %>%
  summarize(
    prop_validated = sum(!is.na(true_spp))/n()
  )
```

`mcmc_sum` *MCMC_sum: A custom function for summarizing MCMC posterior sampling*

Description

MCMC_sum: A custom function for summarizing MCMC posterior sampling

Usage

```
mcmc_sum(out, thin = 1, truth)
```

Arguments

`out`	Draws from a model fit using a probabilistic programming language (e.g., Stan, NIMBLE or JAGS). The expected format of this input is a list, where each entry is a Markov chain.
`thin`	An optional thinning interval.
`truth`	A vector with the true parameter values organized alphanumerically by parameter value (e.g., lambda\[1\], lambda\[2\], psi\[1\], psi\[2\], theta\[1,1\], theta\[1,2\], theta\[2,1\], theta\[2,2\])

Value

A dataframe object summarizing the MCMC draws, including diagnostics, quantiles and posterior means.

Examples

```
# An example fit of one dataset
draws <- ValExp_example_fit

# The data generating values
truth <- c(11,2,0.3, 0.6, 0.9, 0.15, 0.10, 0.85)

mcmc_sum(draws, truth = truth)
```

`plot_bias_vs_calls` *plot_bias_vs_calls: Compare validation designs based on estimation error and expected level of effort*

Description

plot_bias_vs_calls: Compare validation designs based on estimation error and expected level of effort

Usage

```
plot_bias_vs_calls(
  sim_summary,
  calls_summary,
  pars = NULL,
  regex_pars = NULL,
  theta_scenario,
  scenarios,
  max_calls = NULL,
  convergence_threshold = 1.1
)
```

Arguments

`sim_summary`	Simulation summary from many datasets under many validation scenarios in the format output by mcmc_sum.
`calls_summary`	Summary of the validation design in the format as output from summarize_n_validated
`pars`	A character vector of parameters to visualize.
`regex_pars`	String containing the name of a group of parameters to visualize. Must be one of "lambda", "psi", or "theta".
`theta_scenario`	String containing the theta scenario ID that was used to run simulations in run_sims. This string must match between `calls_summary` and `sim_summary`.
`scenarios`	A vector of integers corresponding to the validation designs you would like to visualize.
`max_calls`	The maximum number of calls that can be manually reviewed. All points beyond this threshold will be shaded gray in the plot. Default value is NULL.
`convergence_threshold`	A threshold for the Gelman-Rubin statistic; values below this threshold indicate that a parameter has converged.

Value

A ggplot2 object showing the number of calls validated on the x-axis and the average estimation error on the y-axis.

Examples

```
sim_summary <- example_output
calls_summary <- example_val_sum

plot_bias_vs_calls(sim_summary, calls_summary, regex_pars = "lambda",
                   theta_scenario = "1", scenarios = 1:2,
                   convergence_threshold = 1.05)
```

`plot_coverage_vs_calls` *plot_coverage_vs_calls: Compare validation designs based on coverage of 95% posterior intervals and expected level of effort*

Description

plot_coverage_vs_calls: Compare validation designs based on coverage of 95% posterior intervals and expected level of effort

Usage

```
plot_coverage_vs_calls(
  sim_summary,
  calls_summary,
  pars = NULL,
  regex_pars = NULL,
  theta_scenario,
  scenarios,
  max_calls = NULL,
  convergence_threshold = 1.1
)
```

Arguments

`sim_summary`	Simulation summary from many datasets under many validation scenarios in the format output by mcmc_sum.
`calls_summary`	Summary of the validation design in the format as output from summarize_n_validated
`pars`	A character vector of parameters to visualize.
`regex_pars`	String containing the name of a group of parameters to visualize. Must be one of "lambda", "psi", or "theta".
`theta_scenario`	String containing the theta scenario ID that was used to run simulations in run_sims. This string must match between `calls_summary` and `sim_summary`.
`scenarios`	A vector of integers corresponding to the validation designs you would like to visualize.
`max_calls`	The maximum number of calls that can be manually reviewed. All points beyond this threshold will be shaded gray in the plot. Default value is NULL.
`convergence_threshold`	A threshold for the Gelman-Rubin statistic; values below this threshold (and near 1) indicate that a parameter has converged.

Value

A ggplot2 object showing the number of calls validated on the x-axis and the average coverage of 95% credible intervals on the y-axis.

Examples

```
sim_summary <- example_output
calls_summary <- example_val_sum

plot_coverage_vs_calls(sim_summary, calls_summary, regex_pars = "lambda",
                   theta_scenario = "1", scenarios = 1:2,
                   convergence_threshold = 1.05)
```

`plot_width_vs_calls` *plot_width_vs_calls: Compare validation designs based on 95% posterior interval width and expected level of effort*

Description

plot_width_vs_calls: Compare validation designs based on 95% posterior interval width and expected level of effort

Usage

```
plot_width_vs_calls(
  sim_summary,
  calls_summary,
  pars = NULL,
  regex_pars = NULL,
  theta_scenario,
  scenarios,
  max_calls = NULL,
  convergence_threshold = 1.1
)
```

Arguments

`sim_summary`	Simulation summary from many datasets under many validation scenarios in the format output by mcmc_sum.
`calls_summary`	Summary of the validation design in the format as output from summarize_n_validated
`pars`	A character vector of parameters to visualize.
`regex_pars`	String containing the name of a group of parameters to visualize. Must be one of "lambda", "psi", or "theta".
`theta_scenario`	String containing the theta scenario ID that was used to run simulations in run_sims. This string must match between `calls_summary` and `sim_summary`.
`scenarios`	A vector of integers corresponding to the validation designs you would like to visualize.
`max_calls`	The maximum number of calls that can be manually reviewed. All points beyond this threshold will be shaded gray in the plot. Default value is NULL.

convergence_threshold
: A threshold for the Gelman-Rubin statistic; values below this threshold indicate that a parameter has converged.

Value

A ggplot2 object showing the number of calls validated on the x-axis and the average 95% credible interval on the y-axis.

Examples

```
sim_summary <- example_output
calls_summary <- example_val_sum

plot_width_vs_calls(sim_summary, calls_summary, regex_pars = "lambda",
                    theta_scenario = "1", scenarios = 1:2,
                    convergence_threshold = 1.05)
```

---

run_sims *run_sims: conduct simulations easily*

Description

run_sims: conduct simulations easily

Usage

```
run_sims(
  data_list,
  zeros_list,
  DGVs,
  theta_scenario_id,
  parallel = TRUE,
  niter = 2000,
  nburn = floor(niter/2),
  thin = 1,
  nchains = 3,
  save_fits = FALSE,
  save_individual_summaries_list = FALSE,
  directory = tempdir()
)
```

Arguments

data_list
: nested list of masked dataframes (datasets nested within scenarios – this is the format of `masked_dfs` output from simulate_validatedData)

zeros_list	list of dataframes containing the site/visit/true_spp/id_spp combinations where no calls were observed.
DGVs	A named list with entries psi, lambda and theta containing the true values of the respective parameters.
theta_scenario_id	A character string ID for the simulations being run.
parallel	Should models be fit in parallel? Default value is TRUE.
niter	Number of iterations per MCMC chain.
nburn	Number of warmup iterations.
thin	Thinning interval for the MCMC chains.
nchains	The number of chains.
save_fits	Should individual model fits be saved? This could require large amounts of disk space if you are fitting many large models to big datasets. Default value is FALSE.
save_individual_summaries_list	Should summaries for individual model fits be saved? While this requires much less space than save_fits, we still recommend keeping this at the default value of FALSE. Only use it if you anticipate that simulations may be interrupted.
directory	The directory to save objects. Defaults to tempdir(), but users should specify a permanent location for real simulation studies as tempdir() is cleared at the end of the R session.

Value

a dataframe with the summaries (from mcmc_sum) for all scenarios and datasets. A copy of this output is also saved to the current working directory.

Examples

```
# :::::::::::: Simulate data :::::::::::: #
psi <- c(0.3, 0.6)
lambda <- c(11, 2)

nspecies <- length(psi)
nsites <- 30
nvisits <- 5

test_theta1 <- matrix(c(0.9, 0.1, 0.15, 0.85), byrow = TRUE, nrow = 2)
val_scenarios <- list(spp1 = c(.75, .5), spp2 = .5)

td <- withr::local_tempdir()

fake_data <- simulate_validatedData(
  n_datasets = 5,
  design_type = "BySpecies",
  scenarios = val_scenarios,
  nsites = nsites,
  nvisits = nvisits,
```

```
  nspecies = nspecies,
  psi = psi,
  lambda = lambda,
  theta = test_theta1,
  save_datasets = FALSE,
  save_masked_datasets = FALSE,
  directory = td
)

# :::::::::::::: run simulations on sim'd data :::::::::::: #



out <- run_sims(
  data_list = fake_data$masked_dfs,
  zeros_list = fake_data$zeros,
  DGVs = list(lambda = lambda, psi = psi, theta = test_theta1),
  theta_scenario_id = 'StratBySpecies_1',
  parallel = FALSE,
  nchains = 2,
  niter = 500,
  nburn = 250,
  thin = 1,
  save_fits = FALSE,
  save_individual_summaries_list = FALSE,
  directory = td
)
```

`sim_dat` *Simulate data from the count-detection model with counts per site-visit*

Description

Simulate data from the count-detection model with counts per site-visit

Usage

```
sim_dat(
  nsites = 100,
  nspecies = 8,
  nvisits = 4,
  seed = NULL,
  psi = stats::runif(nspecies, 0.4, 0.9),
  lambda = abs(stats::rnorm(nspecies, 0, 100)),
  theta = t(apply(18 * diag(nspecies) + 2, 1, function(x) nimble::rdirch(1, x)))
)
```

Arguments

nsites	the number of sites assumed in the design. Default value is 100.
nspecies	the number of species in the assemblage. Default is 8.
nvisits	the number of visits (detector nights) assumed for each site. Default is 4.
seed	optional seed if you would like to reproduce the data simulation.
psi	a vector of length nspecies that contains the occurrence probabilities for each species in the assemblage. These values must be in \[0,1\]. Default is to draw a random vector from a U(.4, .9) distribution.
lambda	vector of length nspecies that contains the relative activity parameters for each species. Note these values need to be positive. By default, lambda values are the absolute value of normal(0, 100) random variables.
theta	n nspecies x nspecies matrix containing the (mis)classification probabilities for each species. All entries must be in (0,1], with the rows of the matrix summing to 1. The default draws rows from a dirichlet distribution with concentrations determined by location in the matrix (diagonal values have higher concentrations).

Value

A list containing full_df, a complete dataframe simulated under the user's specified parameter settings. et_sim_datasets.R. The second list element is params, the parameters used to simulate data in list form.

Examples

```
fake_data <- sim_dat(
   nsites = 30,
   nspecies = 2,
   nvisits = 3,
   seed = 101,
   psi = c(.3, .6),
   lambda = c(8, 3)
)

head (fake_data$full_df)
```

---

simulate_validatedData

Simulate many datasets under candidate validation designs

Description

Simulate many datasets under candidate validation designs

Usage

```
simulate_validatedData(
  n_datasets,
  design_type = c("BySpecies", "FixedPercent"),
  scenarios = NULL,
  nsites = 100,
  nspecies = 8,
  nvisits = 3,
  psi = runif(nspecies, 0.1, 0.9),
  lambda = abs(rnorm(nspecies, 0, 5)),
  theta = t(apply(diag(18, nrow = nspecies) + 2, 1, function(x) {

    nimble::rdirch(alpha = x)
 })),
  confirmable_limits = NULL,
  scen_expand = TRUE,
  scen_df = NULL,
  save_datasets = FALSE,
  save_masked_datasets = FALSE,
  directory = tempdir()
)
```

Arguments

n_datasets	The number of datasets you would like to have simulated. Each of these simulated datasets will be subjected to all candidate validation designs.
design_type	Character string, either "BySpecies" for a stratified-by-species design, or "FixedPercentage" for a fixed effort design (see Oram et al., in review for more details on each of these)
scenarios	if `design_type = "BySpecies"`, the `scenarios` argument must be a list with each entry corresponding to the potential levels of effort for a particular autoID label. If `design_type == "FixedPercent"`, then the `scenarios` argument must be a vector with each entry corresponding to a potential percent of calls to be sampled from the first visit at each site. See vignette for an example.
nsites	number of sites in each dataset
nspecies	size of the species assemblage
nvisits	the number of visits to each site. Note that these simulations assume a balanced design.
psi	a vector of length nspecies with the assumed occurrence probabilities for each species
lambda	a vector of length nspecies with the assumed relative activity levels for each species. Make sure the order is correct and matches psi.
theta	a matrix containing the (mis)classification probabilities. The rows of this matrix must sum to 1. See vignette for an example.

`confirmable_limits`
: A numeric vector containing the lower and and upper bounds on the site-visit probabilities that a recording can be validated ("confirmed").

`scen_expand`
: If `design_type = "BySpecies"`, should `simulate_validatedData` expand the list of `scenarios`? If TRUE (the default value), then `scenarios` must be a list; if FALSE, then `simulate_validatedData` expects a user-supplied dataframe object through the `scen_df` argument.

`scen_df`
: If `scen_expand = FALSE`, a user-supplied dataframe object with each row corresponding to the validation scenario and each column to the species. Default value is NULL.

`save_datasets`
: logical. If TRUE, the datasets without any masking of true species labels (i.e., corresponding to complete validation of all recordings) will be saved. Default value is FALSE.

`save_masked_datasets`
: logical. If TRUE, the masked datasets (i.e., the simulated datasets with partial validation according to the simulation scenario) will be saved. This means that there will be n_datasets x nrow(scenarios_dataframe) datasets saved: one for each dataset under each validation scenario. Default value is FALSE.

`directory`
: character. Required if save_datasets = TRUE or save_masked_datasets = TRUE. This is where the datasets will be saved. By default, a temporary directory will be used. This *must* be changed if access to saved datasets is desired after the end of the R session, as tempdir() is cleared at the end of the session.

Value

A list containing three elements:

1. `full_datasets`: A list of length n_datasets with unmasked datasets (i.e., full validation of all recordings). If `save_datasets = TRUE`, then these will be saved individually in `directory` as dataset_n.rds, where n is the dataset number.
2. `zeros`: A list of length n_datasets containing all of the site-visits where no recordings of a certain classification were observed. For example, if, in dataset 10, there were no calls from species 1 that were classified as 3 on visit 4 to site 156, then the 10th entry of this list would contain a dataset with a row corresponding tosite = 156, visit = 4, true_spp = 1, id_spp = 3, with count = 0. These zeros are necessary for housekeeping in the model-fitting process. If `save_datasets = TRUE`, the zeros for each each dataset will be saved in `directory` individually as zeros_in_dataset_n.rds, where n is the dataset number.
3. `masked_dfs`: A nested list containing each dataset masked under each scenario. masked_dfs\[\[9\]\]\[\[27\]\] contains dataset 27, assuming validation scenario 9. If `save_masked_datasets = TRUE`, then each dataset/scenario scenario combination is saved individually in `directory` as dataset_n_masked_under_scenario_s. where n is the dataset number and s is the scenario number.

Examples

```
psi <- c(0.3, 0.6)
lambda <- c(11, 2)

nspecies <- length(psi)
```

```
nsites <- 30
nvisits <- 5

test_theta1 <- matrix(c(0.9, 0.1, 0.15, 0.85), byrow = TRUE, nrow = 2)
val_scenarios <- list(spp1 = c(.75, .5), spp2 = .5)

fake_data <- simulate_validatedData(
  n_datasets = 5,
  design_type = "BySpecies",
  scenarios = val_scenarios,
  nsites = nsites,
  nvisits = nvisits,
  nspecies = nspecies,
  psi = psi,
  lambda = lambda,
  theta = test_theta1,
  save_datasets = FALSE,
  save_masked_datasets = FALSE,
  directory = tempdir()
)
```

`summarize_n_validated` *Summarize the number of validated recordings*

Description

Summarize the number of validated recordings

Usage

```
summarize_n_validated(data_list, scenario_numbers, theta_scenario)
```

Arguments

`data_list`	A list of masked datasets in the format output from simulate_validatedData.
`scenario_numbers`	An integer vector denoting the scenarios to be summarized.
`theta_scenario`	The identifier for the classifier scenario as a character string.

Value

A dataframe object that contains the theta scenario, validation scenario, and number of validated calls.

Examples

```
psi <- c(0.3, 0.6)
lambda <- c(11, 2)

nspecies <- length(psi)
nsites <- 30
nvisits <- 5

test_theta1 <- matrix(c(0.9, 0.1, 0.15, 0.85), byrow = TRUE, nrow = 2)
val_scenarios <- list(spp1 = c(.75, .5), spp2 = .5)

fake_data <- simulate_validatedData(
  n_datasets = 5,
  design_type = "BySpecies",
  scenarios = val_scenarios,
  nsites = nsites,
  nvisits = nvisits,
  nspecies = nspecies,
  psi = psi,
  lambda = lambda,
  theta = test_theta1,
  save_datasets = FALSE,
  save_masked_datasets = FALSE,
  directory = withr::local_tempdir()
)

summarize_n_validated(
  data_list = fake_data$masked_dfs,
  scenario_numbers = 1:nrow(fake_data$scenarios_df),
  theta_scenario = '1'
)
```

---

tune_mcmc *Get suggested MCMC settings prior to starting your simulations*

Description

Get suggested MCMC settings prior to starting your simulations

Usage

```
tune_mcmc(dataset, zeros, return_fit = TRUE)
```

Arguments

dataset	A dataframe containing the validated and ambiguous data to be fit. Expected format is that of a single masked dataframe contained in the output from simulate_validatedData. We recommend using a dataset from your lowest-effort validation scenario.

zeros	A dataframe containing the site/visit/true_spp/id_spp combinations that were never observed (count = 0). This will be one of the elements of the zeros object ouput from simulate_validatedData.
return_fit	A logical indicating whether the draws from the posterior should be returned. Default value is set to TRUE to encourage visual inspection of chains during the tuning process.

Value

A list containing the expected time to fit a single dataset with 10,000 iterations per chain, the draws for each chain, a guess for the minimum number of iterations and warmup required to to ensure Rhat <= 1.1 for all model parameters, and a dataframe containing effective sample sizes for each parameter.

Examples

```
psi <- c(0.3, 0.6)
lambda <- c(11, 2)

nspecies <- length(psi)
nsites <- 30
nvisits <- 5

test_theta1 <- matrix(c(0.9, 0.1, 0.15, 0.85), byrow = TRUE, nrow = 2)
val_scenarios <- list(spp1 = c(.75, .5), spp2 = .5)

fake_data <- simulate_validatedData(
  n_datasets = 5,
  design_type = "BySpecies",
  scenarios = val_scenarios,
  nsites = nsites,
  nvisits = nvisits,
  nspecies = nspecies,
  psi = psi,
  lambda = lambda,
  theta = test_theta1,
  save_datasets = FALSE,
  save_masked_datasets = FALSE,
  directory = withr::local_tempdir()
)
# scenario 1 has the lowest effort, so use the 5th dataset from that scenario
# Not run during checks
if (interactive()) {
# note the index of the zeros matches the index of the dataset
tune_mcmc(dataset = fake_data$masked_dfs[[1]][[5]], zeros = fake_data$zeros[[5]])
}
```

`ValExp_example_fit` *ValExp_example_fit*

Description

Draws from an MCMC algorithm fit to simulated data. This object exists solely for easing examples in documentation of mcmc_sum.

Format

A list of length 3 matrices with columns and rows as described here.

Details

- Columns: Parameters (lambda, psi, and theta for a two species assemblage)
- Rows; Iterations of the MCMC. There are 1000 post-warmup draws for each.

Source

Simulated data using simulate_validatedData, then isolated one dataset and the corresponding zeros. Draws were obtained from model fitting using the code inside tune_mcmc.

Examples

```
str(ValExp_example_fit)
```

`visualize_parameter_group` *visualize_parameter_group*

Description

Visualize simulation results for entire groups of parameters

Usage

```
visualize_parameter_group(
  sim_summary,
  pars,
  theta_scenario,
  scenarios,
  convergence_threshold = 1.1
)
```

Arguments

sim_summary	Summary output in the format of run_sims().
pars	The parameters to be visualized.
theta_scenario	The theta scenario IDs. This should match the theta_scenario_id argument ofrun_sims().
scenarios	Scenarios to be visualized.
convergence_threshold	If the Gelman-Rubin statistic is below this value, consider an MCMC to have converged. Default value is 1.1, but we recommend 1.05.

Value

A ggplot object summarizing simulation results.

Examples

```
visualize_parameter_group(
  example_output,
  pars = "lambda",
  theta_scenario = "1",
  scenarios = 1:2
)
```

---

visualize_single_parameter

visualize_single_parameter

Description

See detailed simulation study results for a parameter of interest

Usage

```
visualize_single_parameter(
  sim_summary,
  par,
  theta_scenario,
  scenarios,
  convergence_threshold = 1.1
)
```

Arguments

`sim_summary`	Summary output in the format of run_sims.
`par`	The parameter to be visualized.
`theta_scenario`	The theta scenario IDs. This should match the `theta_scenario_id` argument of`run_sims()`.
`scenarios`	Scenarios to be visualized.
`convergence_threshold`	If the Gelman-Rubin statistic is below this value, consider an MCMC to have converged. Default value is 1.1, but we recommend 1.05.

Value

A ggplot object summarizing simulation results.

Examples

```
visualize_single_parameter(
  example_output,
  par = "lambda[1]",
  theta_scenario = "1",
  scenarios = 1:2,
  convergence_threshold = 1.05
)
```

Index