Package 'HouseTrans'

February 16, 2026

Title Bayesian Household Transmission Chain Analysis with Viral Load Dynamics

Version 0.0.9

Description Provides a streamlined pipeline to simulate household infection dynamics, estimate transmission parameters, and visualize epidemic timelines. Uses a Bayesian approach with Stan that models transmission probability as a Function of viral load, seasonality and infectivity, multiple infection episodes (reinfections), and waning immunity modeling.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Biarch true

Depends R (>= 3.5.0)

Imports gridExtra,
methods,
Rcpp (>= 0.12.0),
rstan (>= 2.26.0),
rstantools (>= 2.3.0),
dplyr (>= 1.0.0),
tidyr,
tibble,
ggplot2,
data.table,
scales,
rlang,
stats,
utils

Suggests testthat (>= 3.0.0),
knitr,
rmarkdown,
ggpubr

LinkingTo BH (>= 1.66.0),
Rcpp (>= 0.12.0),
RcppEigen (>= 0.3.3.3.0),
RcppParallel (>= 5.0.1),
rstan (>= 2.26.0),
StanHeaders (>= 2.26.0)

SystemRequirements GNU make

VignetteBuilder knitr

LazyData true

Config/testthat/edition 3

URL https://github.com/yirenhou2001/HouseTrans

BugReports https://github.com/yirenhou2001/HouseTrans

Contents

`fit_household_model` *Fit Household Transmission Model*

Description

Fits the compiled Stan model to the prepared household data.

Usage

```
fit_household_model(
  stan_data,
  iter = 2000,
  chains = 4,
  warmup = 1000,
  init_fun = NULL,
  control = list(adapt_delta = 0.95, max_treedepth = 15),
  cores = 4,
  refresh = 50,
  ...
)
```

Arguments

stan_data	A list of data formatted by prepare_stan_data.
iter	Integer; number of iterations per chain (including warmup).
chains	Integer; number of Markov chains.
warmup	Integer; number of warmup iterations.
init_fun	Function or List; initial values for the sampler.
control	List; Stan control parameters.
cores	Integer; number of CPU cores.
refresh	Integer; refresh rate for output.
...	Additional arguments passed to rstan::sampling.

Value

A stanfit object containing the posterior samples.

Examples

```
## Not run:
household_profile <- list(
  prob_adults   = c(0, 0, 1),
  prob_infant   = 1.0,
  prob_siblings = c(0, .8, .2),
  prob_elderly  = c(0.7, 0.1, 0.2)
)

sim_res <- simulate_multiple_households_comm(
  n_households = 100,
  viral_testing = "viral load",
  infectious_shape = 10,
  infectious_scale = 1,
  waning_shape = 6,
  waning_scale = 10,
  surveillance_interval = 4,
  start_date = "2024-07-01",
  end_date = "2025-06-30",
  surveillance_df = surveillance_data,
  seed = 123,
  household_profile_list = household_profile
)
person_covariates <- sim_res$hh_df %>%
  dplyr::select(hh_id, person_id, vacc_status) %>%   dplyr::distinct()

df_for_stan <- sim_res$diagnostic_df %>%
  dplyr::left_join(person_covariates, by = c("hh_id", "person_id"))

stan_input <- prepare_stan_data(
  df_clean = df_for_stan,
  use_vl_data      = 1,
  study_start_date = as.Date(study_start),
  study_end_date   = as.Date(study_end),
  surveillance_df  = surveillance_data,
  study_start_date = as.Date(study_start),
  study_end_date   = as.Date(study_end),
```

```
  imputation_params = VL_params_list)
fit <- fit_household_model(stan_input)

## End(Not run)
```

GenSyn *Simulate Household Transmission and Estimate Parameters*

Description

Simulates household infection dynamics with viral load trajectories, reinfections, and covariate effects, then estimates transmission parameters using Bayesian inference via Stan.

Usage

```
GenSyn(
  n_households = 50,
  start_date = "2024-07-01",
  end_date = "2025-06-30",
  surveillance_df = NULL,
  seasonal_forcing_list = NULL,
  covariates_config = NULL,
  alpha_comm_by_role = 5e-04,
  beta1 = 0.008,
  beta2 = 0.008,
  delta = 0,
  phi_by_role = c(adult = 1, infant = 4, toddler = 5, elderly = 1),
  kappa_by_role = c(adult = 1, infant = 1, toddler = 1.2, elderly = 1),
  infectious_shape = 3,
  infectious_scale = 1,
  waning_shape = 16,
  waning_scale = 10,
  peak_day = 1,
  width = 4,
  detect_threshold_log10 = 1e-06,
  detect_threshold_Ct = 99,
  surveillance_interval = 1,
  test_daily = FALSE,
  viral_testing = "viral load",
  V_ref = 3,
  V_rho = 2.5,
  Ct_50 = 40,
  Ct_delta = 2,
  VL_params_list = NULL,
  Ct_params_list = NULL,
  household_profile_list = NULL,
  max_infections = Inf,
  use_vl_data = TRUE,
  priors = NULL,
  covariates_susceptibility = NULL,
  covariates_infectivity = NULL,
```

```
  recovery_params = NULL,
  imputation_params = NULL,
  stan_chains = 4,
  stan_iter = 2000,
  stan_warmup = 1000,
  stan_control = list(adapt_delta = 0.95, max_treedepth = 15),
  stan_cores = 4,
  stan_refresh = 50,
  seed = NULL
)
```

Arguments

`n_households`	Integer; number of households to simulate.
`start_date`, `end_date`	Character or Date; study period bounds.
`surveillance_df`	Optional data frame with 'date' and 'cases' columns for seasonal forcing. If provided, overrides `seasonal_forcing_list`.
`seasonal_forcing_list`	Optional named list with seasonal forcing vectors for each role (adult, infant, toddler, elderly).
`covariates_config`	Optional list of covariate configurations for simulation. Each element should be a list with: `name` (column name), `efficacy` (effect size, 0-1), `effect_on` ("susceptibility", "infectivity", or "both"), and `coverage` (list of probabilities by role).
`alpha_comm_by_role`	Numeric; baseline community acquisition rate.
`beta1`, `beta2`	Numeric; transmission coefficients.
`delta`	Numeric; household size scaling exponent.
`phi_by_role`	Named numeric vector; susceptibility multipliers by role.
`kappa_by_role`	Named numeric vector; infectivity multipliers by role.
`infectious_shape`, `infectious_scale`	Numeric; gamma parameters for infectious period.
`waning_shape`, `waning_scale`	Numeric; gamma parameters for immunity waning period.
`peak_day`, `width`	Numeric; infectivity profile parameters.
`detect_threshold_log10`	Numeric; viral load detection threshold (log10).
`detect_threshold_Ct`	Numeric; Ct value detection threshold.
`surveillance_interval`	Integer; days between scheduled tests.
`test_daily`	Logical; switch to daily testing after first detection.
`viral_testing`	Character; `"viral load"` or `"Ct"`.
`V_ref`, `V_rho`	Numeric; viral load reference and power for transmission.
`Ct_50`, `Ct_delta`	Numeric; Ct-based infectivity parameters.

`VL_params_list` Named list; role-specific viral load trajectory parameters.

`Ct_params_list` Named list; role-specific Ct trajectory parameters.

`household_profile_list`
Named list; household composition probabilities.

`max_infections` Integer; maximum infections per person (for reinfection modeling).

`use_vl_data` Logical; whether to use viral load data in estimation.

`priors` Named list; prior specifications for Stan model. Each element should be a list with `dist` ("normal", "uniform", "lognormal") and `params` (parameter vector). Available priors: beta1, beta2, alpha, covariates, gen_shape, gen_rate, ct50, slope.

`covariates_susceptibility`
Character vector; names of susceptibility covariates.

`covariates_infectivity`
Character vector; names of infectivity covariates.

`recovery_params`
Named list; Gamma parameters (shape, scale) for immunity tail duration by role.

`imputation_params`
Named list; parameters for viral curve imputation by role.

`stan_chains`, `stan_iter`, `stan_warmup`
Integers; Stan sampling controls.

`stan_control` List; Stan control parameters.

`stan_cores` Integer; number of CPU cores for Stan.

`stan_refresh` Integer; refresh rate for Stan output.

`seed` Integer; random seed for reproducibility.

Value

An object of class `"GenSynResult"` containing:

$call The matched call

$n_households Number of households simulated

$simulation Raw simulation output with hh_df and diagnostic_df

$stan_data Data prepared for Stan

$fit The stanfit object

$postprocessing Tidy posterior summary

$attack_rates Attack rate and reinfection summaries

$transmission_chains Reconstructed transmission links

See Also

`TransmissionChainAnalysis`, `plot.GenSynResult`

Examples

```
## Not run:
# Basic simulation
result <- GenSyn(
  n_households = 50,
  start_date = "2024-07-01",
  end_date = "2025-06-30",
  stan_chains = 2,
  stan_iter = 1000,
  stan_warmup = 500,
  seed = 123
)

# View results
print(result)
plot(result, which = "posterior")

## End(Not run)
```

`plot.GenSynResult` *Plot method for GenSynResult*

Description

Plot method for GenSynResult

Usage

```
## S3 method for class 'GenSynResult'
plot(x, which = "posterior", print = TRUE, hh_id = 1, prob_cutoff = 0, ...)
```

Arguments

`x`	A `GenSynResult` object.
`which`	Character vector specifying which plots to generate. Options: `"posterior"`, `"covariate_effects"`, `"epidemic_curve"`, `"transmission_chains"`, or `"all"`.
`print`	Logical; whether to print plots immediately.
`hh_id`	Integer; household ID for transmission chain plot (default: 1).
`prob_cutoff`	Numeric; minimum probability threshold for chain links (default: 0).
`...`	Additional arguments (unused).

Value

A named list of ggplot objects (invisibly if `print = TRUE`).

Examples

```
## Not run:
# Basic simulation
result <- GenSyn(
  n_households = 50,
  start_date = "2024-07-01",
  end_date = "2025-06-30",
  stan_chains = 2,
  stan_iter = 1000,
  stan_warmup = 500,
  seed = 123
)

# View results
plot(result, which = "posterior")

## End(Not run)
```

`plot.TransmissionChainResult` *Plot method for TransmissionChainResult*

Description

Plot method for TransmissionChainResult

Usage

```
## S3 method for class 'TransmissionChainResult'
plot(
  x,
  which = "posterior",
  print = TRUE,
  hh_id = 1,
  prob_cutoff = 0,
  bin_width = 7,
  ...
)
```

Arguments

`x`	A `TransmissionChainResult` object.
`which`	Character vector specifying which plots to generate. Options: `"posterior"`, `"transmission_chains"`, `"covariate_effects"`, `"epidemic_curve"`, or `"all"`.
`print`	Logical; whether to print plots immediately.
`hh_id`	Integer; household ID for transmission chain plot (default: 1).
`prob_cutoff`	Numeric; minimum probability threshold for chain links (default: 0).
`bin_width`	Integer; number of days per bin for epidemic curve (default: 7).
`...`	Additional arguments (unused).

Value

A named list of ggplot objects (invisibly if `print = TRUE`).

Examples

```
## Not run:
# Per-person episode format
T_max <- 30
df_person <- data.frame(
  hh_id = c("HH1","HH1","HH1","HH2","HH2","HH2"),
  person_id = c(1, 2, 3, 1, 2, 3),
  role = c("adult","infant","elderly","adult","infant","elderly"),
  infection_time = c(2, 4, NA, 1, 3, NA),
  infectious_start = c(3, 6, NA, 2, 5, NA),
  infectious_end = c(8, 9, NA, 7, 9, NA),
  infection_resolved = c(9, 10, NA, 8, 10, NA)
)

result <- TransmissionChainAnalysis(
  user_data = df_person,
  max_days = T_max,
  stan_chains = 2,
  stan_iter = 1000,
  stan_warmup = 500
)

plot(result, which = "posterior")

## End(Not run)
```

---

plot_covariate_effects

Plot Covariate Coefficients (Forest Plot)

Description

Creates a forest plot showing the effects of covariates on susceptibility and infectivity.

Usage

```
plot_covariate_effects(fit, stan_data)
```

Arguments

`fit`	A `stanfit` object.
`stan_data`	The list data passed to Stan.

Value

A ggplot object.

Examples

```
## Not run:
# fit <- run_household_stan(stan_data = stan_input, ...)
# stan_input should include K_susc / K_inf and match the fitted model
p_beta <- plot_covariate_effects(fit, stan_input)
print(p_beta)

## End(Not run)
```

`plot_epidemic_curve` *Plot Dual-Axis Epidemic Curve (Binned)*

Description

Overlays a stacked bar chart of simulated infections with a line chart of surveillance data. Both datasets are aggregated by 'bin_width' days.

Usage

```
plot_epidemic_curve(
  sim_result,
  surveillance_df,
  start_date_str = "2024-07-01",
  bin_width = 7
)
```

Arguments

`sim_result`	Output object from `simulate_multiple_households_comm`.
`surveillance_df`	Dataframe with `date` and `cases` columns.
`start_date_str`	Character; start date of the simulation.
`bin_width`	Integer; aggregation window in days (default 7).

Value

A ggplot object.

Examples

```
## Not run:
# sim_result should be a simulation output containing sim_result$hh_df
# with at least: infection_time, role
# surveillance_df must include: date, cases

p_epi <- plot_epidemic_curve(
  sim_result,
  surveillance_df,
  start_date_str = "2024-07-01",
  bin_width = 7
)
print(p_epi)
```

```
## End(Not run)
```

`plot_household_timeline` *Plot Household Timeline with Transmission Chains*

Description

Creates a timeline visualization of infections within a household, showing transmission links with probabilities.

Usage

```
plot_household_timeline(
  trans_df,
  stan_data,
  target_hh_id,
  start_date_str = "2024-10-08",
  prob_cutoff = 0,
  plot_width = 11,
  plot_height = 7
)
```

Arguments

`trans_df`	Data frame from `reconstruct_transmission_chains`.
`stan_data`	The list data passed to Stan.
`target_hh_id`	Integer; household ID to plot.
`start_date_str`	Character; start date string.
`prob_cutoff`	Numeric; minimum probability for displaying links.
`plot_width`, `plot_height`	Numeric; plot dimensions.

Value

A ggplot object.

Examples

```
## Not run:
# After fitting the Stan model and post-processing (object names may vary):
# fit       <- run_household_stan(stan_data = stan_input, ...)
# chains    <- postprocess_stan_fit(fit, stan_input)
# trans_df  <- chains$trans_df   # must include: hh_id, source, target, prob

# Plot a specific household (e.g., HH #2)
p_hh <- plot_household_timeline(
  trans_df,
  stan_input,
  target_hh_id = 2,
```

```
  start_date_str = "2024-10-08",
  prob_cutoff = 0.10
)
print(p_hh)

## End(Not run)
```

`plot_posterior_distributions` *Plot Posterior Distributions (Phi and Kappa)*

Description

Creates violin plots of posterior distributions for susceptibility and infectivity parameters by role.

Usage

```
plot_posterior_distributions(fit)
```

Arguments

`fit`	A `stanfit` object.

Value

A combined ggplot object.

Examples

```
## Not run:
# fit <- run_household_stan(stan_data = stan_input, ...)
p_post <- plot_posterior_distributions(fit)
print(p_post)

## End(Not run)
```

`plot_user_epidemic_curve` *Plot Epidemic Curve for User Data*

Description

Creates an epidemic curve from user-provided data, optionally overlaying surveillance data if available.

Usage

```
plot_user_epidemic_curve(
  user_data,
  start_date,
  surveillance_df = NULL,
  bin_width = 7
)
```

Arguments

user_data	Dataframe with user's infection data.
start_date	Date; study start date.
surveillance_df	Optional dataframe with date and cases columns.
bin_width	Integer; aggregation window in days (default 7).

Value

A ggplot object.

Examples

```
## Not run:
# user_data can be in per-person format (recommended for this example),
# with at least: infection_time, role (and optionally HH, individual_ID, etc.)

p_user <- plot_user_epidemic_curve(
  user_data,
  start_date = "2024-10-08",
  bin_width = 7
)
print(p_user)

# Optional: overlay surveillance (must include: date, cases)
p_user2 <- plot_user_epidemic_curve(
  user_data,
  start_date = "2024-10-08",
  surveillance_df = surveillance_df,
  bin_width = 7
)
print(p_user2)

## End(Not run)
```

---

prepare_stan_data *Prepare Data for Stan Model*

Description

Prepares household observation data for Stan model fitting, including imputation of episode timing, covariate matrices, and viral load data.

Usage

```
prepare_stan_data(
  df_clean,
  surveillance_df = NULL,
  role_levels = c("adult", "infant", "toddler", "elderly"),
  study_start_date = as.Date("2024-07-01"),
  study_end_date = as.Date("2025-07-01"),
  seasonal_forcing_list = NULL,
```

```
  use_vl_data = TRUE,
  covariates_susceptibility = NULL,
  covariates_infectivity = NULL,
  recovery_params = NULL,
  imputation_params = NULL,
  priors = list(),
  seed = 123
)
```

Arguments

df_clean	Dataframe with observation data (must contain 'episode_id').
surveillance_df	Dataframe with columns 'date' and 'cases'.
role_levels	Character vector; canonical role levels.
study_start_date, study_end_date	Date objects; study period.
seasonal_forcing_list	Named list of forcing vectors.
use_vl_data	Logical; whether to use viral load data.
covariates_susceptibility	Character vector; susceptibility covariate names.
covariates_infectivity	Character vector; infectivity covariate names.
recovery_params	List of Gamma parameters (shape, scale) by role.
imputation_params	List of viral curve parameters by role.
priors	List of flexible priors (dist, params).
seed	Integer; random seed.

Value

A list of data formatted for Stan.

Examples

```
## Not run:
household_profile <- list(
  prob_adults   = c(0, 0, 1),
  prob_infant   = 1.0,
  prob_siblings = c(0, .8, .2),
  prob_elderly  = c(0.7, 0.1, 0.2)
)

sim_res <- simulate_multiple_households_comm(
  n_households = 100,
  viral_testing = "viral load",
  infectious_shape = 10,
  infectious_scale = 1,
  waning_shape = 6,
  waning_scale = 10,
```

```
  surveillance_interval = 4,
  start_date = "2024-07-01",
  end_date = "2025-06-30",
  surveillance_df = surveillance_data,
  seed = 123,
  household_profile_list = household_profile
)
person_covariates <- sim_res$hh_df %>%
  dplyr::select(hh_id, person_id, vacc_status) %>%   dplyr::distinct()

df_for_stan <- sim_res$diagnostic_df %>%
  dplyr::left_join(person_covariates, by = c("hh_id", "person_id"))

stan_input <- prepare_stan_data(
  df_clean = df_for_stan,
  use_vl_data      = 1,
  study_start_date = as.Date(study_start),
  study_end_date   = as.Date(study_end),
  surveillance_df  = surveillance_data,
  study_start_date = as.Date(study_start),
  study_end_date   = as.Date(study_end),
  imputation_params = VL_params_list)

## End(Not run)
```

`print.GenSynResult` *Print method for GenSynResult*

Description

Print method for GenSynResult

Usage

```
## S3 method for class 'GenSynResult'
print(x, ...)
```

Arguments

`x`	A `GenSynResult` object.
`...`	Unused.

Value

`x`, invisibly.

Examples

```
## Not run:
# Basic simulation
result <- GenSyn(
  n_households = 50,
  start_date = "2024-07-01",
  end_date = "2025-06-30",
```

```
  stan_chains = 2,
  stan_iter = 1000,
  stan_warmup = 500,
  seed = 123
)

# View results
print(result)

## End(Not run)
```

`print.TransmissionChainResult` *Print method for TransmissionChainResult*

Description

Print method for TransmissionChainResult

Usage

```
## S3 method for class 'TransmissionChainResult'
print(x, ...)
```

Arguments

`x`	A `TransmissionChainResult` object.
`...`	Unused.

Value

`x`, invisibly.

Examples

```
## Not run:
# Per-person episode format
T_max <- 30
df_person <- data.frame(
  hh_id = c("HH1","HH1","HH1","HH2","HH2","HH2"),
  person_id = c(1, 2, 3, 1, 2, 3),
  role = c("adult","infant","elderly","adult","infant","elderly"),
  infection_time = c(2, 4, NA, 1, 3, NA),
  infectious_start = c(3, 6, NA, 2, 5, NA),
  infectious_end = c(8, 9, NA, 7, 9, NA),
  infection_resolved = c(9, 10, NA, 8, 10, NA)
)

result <- TransmissionChainAnalysis(
  user_data = df_person,
  max_days = T_max,
  stan_chains = 2,
  stan_iter = 1000,
  stan_warmup = 500
```

```
)

print(result)

## End(Not run)
```

`reconstruct_transmission_chains` *Reconstruct Transmission Chains (Covariate-Aware)*

Description

Identifies potential infectors for each infection event based on posterior estimates, accounting for covariates if present.

Usage

```
reconstruct_transmission_chains(fit, stan_data, min_prob_threshold = 0.01)
```

Arguments

`fit`	A stanfit object.
`stan_data`	The list data passed to Stan.
`min_prob_threshold`	Minimum probability threshold for links.

Value

A data frame of transmission links with probabilities.

Examples

```
## Not run:
household_profile <- list(
  prob_adults   = c(0, 0, 1),
  prob_infant   = 1.0,
  prob_siblings = c(0, .8, .2),
  prob_elderly  = c(0.7, 0.1, 0.2)
)

sim_res <- simulate_multiple_households_comm(
  n_households = 100,
  viral_testing = "viral load",
  infectious_shape = 10,
  infectious_scale = 1,
  waning_shape = 6,
  waning_scale = 10,
  surveillance_interval = 4,
  start_date = "2024-07-01",
  end_date = "2025-06-30",
  surveillance_df = surveillance_data,
  seed = 123,
  household_profile_list = household_profile
```

```
)
person_covariates <- sim_res$hh_df %>%
  dplyr::select(hh_id, person_id, vacc_status) %>%   dplyr::distinct()

df_for_stan <- sim_res$diagnostic_df %>%
  dplyr::left_join(person_covariates, by = c("hh_id", "person_id"))

stan_input <- prepare_stan_data(
  df_clean = df_for_stan,
  use_vl_data      = 1,
  study_start_date = as.Date(study_start),
  study_end_date   = as.Date(study_end),
  surveillance_df  = surveillance_data,
  study_start_date = as.Date(study_start),
  study_end_date   = as.Date(study_end),
  imputation_params = VL_params_list)
fit <- fit_household_model(stan_input)
chains <- reconstruct_transmission_chains(fit, stan_input, min_prob_threshold = 0.001)

## End(Not run)
```

`simulate_multiple_households_comm` *Simulate Multiple Households with Community Transmission*

Description

Simulates household infection dynamics across multiple households with support for generic covariates, reinfections, and viral load trajectories.

Usage

```
simulate_multiple_households_comm(
  n_households = 50,
  surveillance_df = NULL,
  start_date = "2024-07-01",
  end_date = "2025-06-30",
  alpha_comm_by_role = 5e-04,
  beta1 = 0.008,
  beta2 = 0.008,
  delta = 0,
  phi_by_role = c(adult = 1, infant = 4, toddler = 5, elderly = 1),
  kappa_by_role = c(adult = 1, infant = 1, toddler = 1.2, elderly = 1),
  infectious_shape = 3,
  infectious_scale = 1,
  waning_shape = 16,
  waning_scale = 10,
  peak_day = 1,
  width = 4,
  verbose = FALSE,
  seasonal_forcing_list = NULL,
  detect_threshold_log10 = 1e-06,
```

```
  detect_threshold_Ct = 99,
  surveillance_interval = 1,
  test_daily = FALSE,
  viral_testing = "viral load",
  V_ref = 3,
  V_rho = 2.5,
  Ct_50 = 40,
  Ct_delta = 2,
  VL_params_list = NULL,
  Ct_params_list = NULL,
  household_profile_list = NULL,
  perfect_detection = TRUE,
  contact_mat = NULL,
  covariates_config = NULL,
  seed = NULL,
  max_infections = Inf
)
```

Arguments

`n_households`	Integer; number of households to simulate.
`surveillance_df`	Optional data frame with 'date' and 'cases' columns.
`start_date`, `end_date`	Character or Date; study period bounds.
`alpha_comm_by_role`	Numeric; community acquisition rate.
`beta1`, `beta2`	Numeric; transmission coefficients.
`delta`	Numeric; household size scaling exponent.
`phi_by_role`, `kappa_by_role`	Named numeric vectors.
`infectious_shape`, `infectious_scale`	Numeric; Gamma parameters.
`waning_shape`, `waning_scale`	Numeric; Gamma parameters.
`peak_day`, `width`	Numeric; infectivity profile.
`verbose`	Logical; print progress.
`seasonal_forcing_list`	Named list of forcing vectors.
`detect_threshold_log10`	Numeric; detection threshold (log10 VL).
`detect_threshold_Ct`	Numeric; detection threshold (Ct).
`surveillance_interval`	Integer; days between tests.
`test_daily`	Logical; test daily after detection.
`viral_testing`	Character; "viral load" or "Ct".
`V_ref`, `V_rho`	Numeric; VL parameters.

`Ct_50`, `Ct_delta` Numeric; Ct parameters.

`VL_params_list`, `Ct_params_list`
Named lists of trajectory parameters.

`household_profile_list`
Named list for household composition.

`perfect_detection`
Logical.

`contact_mat` Optional contact matrix.

`covariates_config`
List of covariate configurations.

`seed` Integer; random seed.

`max_infections` Integer; max infections per person.

Value

List with hh_df and diagnostic_df.

Examples

```
## Not run:
household_profile <- list(
  prob_adults   = c(0, 0, 1),
  prob_infant   = 1.0,
  prob_siblings = c(0, .8, .2),
  prob_elderly  = c(0.7, 0.1, 0.2)
)

sim_res <- simulate_multiple_households_comm(
  n_households = 100,
  viral_testing = "viral load",
  infectious_shape = 10,
  infectious_scale = 1,
  waning_shape = 6,
  waning_scale = 10,
  surveillance_interval = 4,
  start_date = "2024-07-01",
  end_date = "2025-06-30",
  surveillance_df = surveillance_data,
  seed = 123,
  household_profile_list = household_profile
)

## End(Not run)
```

---

summarize_attack_rates

Summarize Attack Rates & Reinfections

Description

Calculates the Primary Attack Rate (proportion of people infected at least once) and a separate summary of Reinfections (secondary episodes).

Usage

```
summarize_attack_rates(sim_result)
```

Arguments

sim_result	A list object returned by simulate_multiple_households_comm.

Value

A list containing four dataframes:

primary_overall Overall primary attack rate statistics.

primary_by_role Primary attack rate stratified by age group.

reinf_overall Overall summary of reinfection counts and rates.

reinf_by_role Reinfection counts stratified by age group.

Examples

```
## Not run:
household_profile <- list(
  prob_adults   = c(0, 0, 1),
  prob_infant   = 1.0,
  prob_siblings = c(0, .8, .2),
  prob_elderly  = c(0.7, 0.1, 0.2)
)

sim_res <- simulate_multiple_households_comm(
  n_households = 100,
  viral_testing = "viral load",
  infectious_shape = 10,
  infectious_scale = 1,
  waning_shape = 6,
  waning_scale = 10,
  surveillance_interval = 4,
  start_date = "2024-07-01",
  end_date = "2025-06-30",
  surveillance_df = surveillance_data,
  seed = 123,
  household_profile_list = household_profile
)
rates <- summarize_attack_rates(sim_res)

## End(Not run)
```

TransmissionChainAnalysis

Analyze User-Supplied Household Transmission Data

Description

Runs Bayesian estimation on user-provided household testing or episode data. This function does not simulate data; it expects input in tabular form.

Usage

```
TransmissionChainAnalysis(
  user_data,
  start_date = as.Date("2024-01-01"),
  end_date = as.Date("2024-12-31"),
  surveillance_df = NULL,
  seasonal_forcing_list = NULL,
  max_days = NULL,
  use_vl_data = TRUE,
  priors = NULL,
  covariates_susceptibility = NULL,
  covariates_infectivity = NULL,
  recovery_params = NULL,
  imputation_params = NULL,
  role_levels = c("adult", "infant", "toddler", "elderly"),
  stan_chains = 4,
  stan_iter = 2000,
  stan_warmup = 1000,
  stan_control = list(adapt_delta = 0.95, max_treedepth = 15),
  stan_cores = 4,
  stan_refresh = 50,
  seed = NULL
)
```

Arguments

`user_data` A data.frame or list of data.frames with household data. Accepts two formats:

- **Long testing table**: columns `HH`, `individual_ID`, `role`, `test_date`, `infection_status` (and optionally a viral load column)
- **Per-person episode table**: columns `hh_id`, `person_id`, `role`, `infection_time`, `infectious_start`, `infectious_end`, `infection_resolved`

`start_date`, `end_date`
Date objects or character; study period bounds.

`surveillance_df`
Optional data frame with 'date' and 'cases' columns for seasonal forcing.

`seasonal_forcing_list`
Optional named list with seasonal forcing vectors for each role.

`max_days` Integer; maximum time horizon (days).

`use_vl_data` Logical; whether to use viral load data in estimation.

`priors` Named list; prior specifications for Stan model. Each element should be a list with `dist` ("normal", "uniform", "lognormal") and `params` (parameter vector).

`covariates_susceptibility`
Character vector; column names for susceptibility covariates in the data.

`covariates_infectivity`
Character vector; column names for infectivity covariates in the data.

`recovery_params`
Named list; Gamma parameters (shape, scale) for immunity tail duration by role.

`imputation_params`
Named list; parameters for viral curve imputation by role.

role_levels	Character vector; canonical role levels for normalization.
stan_chains, stan_iter, stan_warmup	Integers; Stan sampling controls.
stan_control	List; Stan control parameters.
stan_cores	Integer; number of CPU cores for Stan.
stan_refresh	Integer; refresh rate for Stan output.
seed	Integer; random seed for reproducibility.

Value

An object of class "TransmissionChainResult" containing:

$call The matched call

$user_data The processed user data

$stan_data Data prepared for Stan

$fit The stanfit object

$postprocessing Tidy posterior summary

$transmission_chains Reconstructed transmission links

See Also

GenSyn, plot.TransmissionChainResult

Examples

```
## Not run:
# Per-person episode format
T_max <- 30
df_person <- data.frame(
  hh_id = c("HH1","HH1","HH1","HH2","HH2","HH2"),
  person_id = c(1, 2, 3, 1, 2, 3),
  role = c("adult","infant","elderly","adult","infant","elderly"),
  infection_time = c(2, 4, NA, 1, 3, NA),
  infectious_start = c(3, 6, NA, 2, 5, NA),
  infectious_end = c(8, 9, NA, 7, 9, NA),
  infection_resolved = c(9, 10, NA, 8, 10, NA)
)

result <- TransmissionChainAnalysis(
  user_data = df_person,
  max_days = T_max,
  stan_chains = 2,
  stan_iter = 1000,
  stan_warmup = 500
)

print(result)
plot(result, which = "posterior")

## End(Not run)
```

Index