Package 'Household.Transmission.Chain.Data.Analysis'

February 2, 2026

Title Household Transmission Chain Simulation and Estimation

Version 0.0.0.9000

Description Simulates and analyzes household transmission chains incorporating explicit viral load dynamics. The package provides three main capabilities. (1) Synthetic data generation with role-specific susceptibility and infectivity, community acquisition, within-household transmission driven by time-varying viral loads, and configurable surveillance testing; (2) Data preprocessing that constructs person-day panels and test-time matrices suitable for Bayesian inference, including time-since-infection tracking for latent period modeling; (3) Parameter estimation via Stan-based Markov chain Monte Carlo for community and household transmission rates, with support for estimating latent period duration. Includes diagnostic plots for infection timelines, secondary attack rates stratified by index case viral load, and aggregated incidence summaries. Designed for respiratory pathogens where transmission probability depends on the infector's viral shedding trajectory.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports data.table,
dplyr,
ggplot2,
tibble,
tidyr,
utils,
rstan,
stats

Suggests testthat (>= 3.0.0),
knitr,
rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

Contents

`build_stan_household_arrays` *Build Stan data arrays from household list*

Description

Converts a list of per-household data frames into Stan-ready arrays.

Usage

```
build_stan_household_arrays(
  households,
  T_max = 365L,
  seasonal_forcing_list = NULL,
  alpha_comm_by_role = 0.005,
  beta1 = 0.2,
  beta2 = 0.6,
  V_ref = 1000,
  reference_phi = 1,
  reference_kappa = 1,
  g_peak_day = 2,
  g_width = 4
)
```

Arguments

`households`	List of per-household data frames.
`T_max`	Integer; maximum time horizon.
`seasonal_forcing_list`	Named list of seasonal forcing vectors.
`alpha_comm_by_role`, `beta1`, `beta2`, `V_ref`	Numeric; model parameters.
`reference_phi`, `reference_kappa`	Numeric; reference parameters.
`g_peak_day`, `g_width`	Numeric; infectivity profile parameters.

Value

Named list for Stan.

GenSyn *GenSyn: household simulation & estimation wrapper*

Description

Runs `main_parameter_estimation_pipeline()` to simulate household data, fit via Bayesian Stan, and assemble a `"GenSynResult"`.

Usage

```
GenSyn(
  n_households = 50,
  plots = c("daily", "weekly", "timeline", "sar"),
  print_plots = FALSE,
  index_vl_column = "vl_test",
  start_date = as.Date("2024-01-01"),
  end_date = as.Date("2024-12-31"),
  seasonal_forcing_list = NULL,
  max_days = 365,
  alpha_comm_by_role = 0.005,
  beta1 = 0.3,
  beta2 = 0.05,
  delta = 0,
  phi_by_role = c(adult = 1, child = 7, toddler = 7, elderly = 4),
  kappa_by_role = c(adult = 1, child = 1.5, toddler = 1.5, elderly = 1),
  latent_shape = 3,
  latent_scale = 0.5,
  infectious_shape = 3,
  infectious_scale = 1,
  resolve_shape = 1.5,
  resolve_scale = 0.5,
  peak_day = 1,
  width = 4,
  ptrans_threshold = 0.5,
  detect_threshold_log10 = 1,
  detect_threshold_Ct = 40,
  surveillance_interval = 7,
  test_daily = FALSE,
  viral_testing = "viral load",
  V_ref = 3,
  V_rho = 2.5,
  Ct_50 = 40,
  Ct_delta = 2,
  VL_params_list = default_VL_params,
  Ct_params_list = default_Ct_params,
  household_profile_list = default_household_profile,
  stan_file = "model.stan",
```

```
  stan_chains = 4,
  stan_iter = 2000,
  stan_warmup = 1000,
  stan_control = list(adapt_delta = 0.99, max_treedepth = 20),
  stan_init = "random",
  stan_refresh = 50,
  stan_cores = 4
)
```

Arguments

`n_households`	Integer; number of households to simulate.
`plots`	Character vector of plot names (`"daily"`, `"weekly"`, `"sar"`, `"timeline"`) or `"all"`.
`print_plots`	Logical; print plots if `TRUE`.
`index_vl_column`	Character; viral-load (VL) column name for plotting (default `"vl_test"`).
`start_date`, `end_date`	`Date`; study window.
`seasonal_forcing_list`	Optional named list of role vectors; seasonal forcing.
`max_days`	Integer; maximum simulated days.
`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.
`latent_shape`, `latent_scale`	Numeric; gamma parameters for latent period.
`infectious_shape`, `infectious_scale`	Numeric; gamma parameters for infectious period.
`resolve_shape`, `resolve_scale`	Numeric; gamma parameters for resolution period.
`peak_day`, `width`	Numeric; infectivity profile parameters.
`ptrans_threshold`	Numeric; transmission potential threshold.
`detect_threshold_log10`	Numeric; VL detection threshold in log10.
`detect_threshold_Ct`	Numeric; Ct 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.
`Ct_50`, `Ct_delta`	Numeric; Ct-based infectivity parameters.

`VL_params_list` Named list; role-specific VL trajectory parameters.

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

`household_profile_list`
Named list; household composition probabilities.

`stan_file` Path to a Stan model file.

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

`stan_control` List; Stan `control` list.

`stan_init` Character or function; Stan initialization.

`stan_refresh` Integer; Stan refresh rate.

`stan_cores` Integer; CPU cores for Stan.

Value

A `"GenSynResult"` list with elements: `$call`, `$n_households`, `$results`, `$postprocessing`, `$plot_list`.

See Also

`main_parameter_estimation_pipeline`, `TransmissionChainAnalysis`

Examples

```
## Not run:
# Simulate and estimate
seasonal_forcing_list <- list(
  adult=rep(0.1,365), child=rep(0.1,365), elderly=rep(0.1,365), toddler=rep(0.1,365)
)
fit <- GenSyn(
  n_households=50,
  seasonal_forcing_list=seasonal_forcing_list, max_days=365,
  stan_chains=4, stan_iter=2000, stan_warmup=1000, stan_cores=4
)

## End(Not run)
```

`main_parameter_estimation_pipeline` *Main parameter estimation pipeline*

Description

End-to-end workflow for household transmission estimation via Bayesian Stan. Works with simulated or user-provided data.

Usage

```
main_parameter_estimation_pipeline(
  user_data = NULL,
  synthetic_data = TRUE,
  n_households = 10,
  start_date = as.Date("2024-01-01"),
  end_date = as.Date("2024-12-31"),
  seasonal_forcing_list = NULL,
  max_days = 365,
  alpha_comm_by_role = 0.005,
  beta1 = 0.3,
  beta2 = 0.05,
  delta = 0,
  phi_by_role = c(adult = 1, child = 7, toddler = 7, elderly = 4),
  kappa_by_role = c(adult = 1, child = 1.5, toddler = 1.5, elderly = 1),
  latent_shape = 3,
  latent_scale = 0.5,
  infectious_shape = 3,
  infectious_scale = 1,
  resolve_shape = 1.5,
  resolve_scale = 0.5,
  peak_day = 1,
  width = 4,
  ptrans_threshold = 0.5,
  detect_threshold_log10 = 1,
  detect_threshold_Ct = 40,
  surveillance_interval = 7,
  test_daily = FALSE,
  viral_testing = "viral load",
  V_ref = 3,
  V_rho = 2.5,
  Ct_50 = 40,
  Ct_delta = 2,
  VL_params_list = default_VL_params,
  Ct_params_list = default_Ct_params,
  household_profile_list = default_household_profile,
  stan_file = "model.stan",
  stan_chains = 4,
  stan_iter = 2000,
  stan_warmup = 1000,
  stan_control = list(adapt_delta = 0.99, max_treedepth = 20),
  stan_init = "random",
  stan_refresh = 50,
  stan_cores = 4
)
```

Arguments

user_data	Optional; data.frame or list of data.frames.
synthetic_data	Logical; if TRUE, simulate data internally.
n_households	Integer; number of households to simulate.

start_date, end_date
: Date; analysis window.

seasonal_forcing_list
: Optional named list for seasonality.

max_days
: Integer; maximum days.

alpha_comm_by_role
: Numeric; community acquisition rate.

beta1, beta2
: Numeric; transmission coefficients.

delta
: Numeric; household size scaling.

phi_by_role, kappa_by_role
: Named numeric vectors.

latent_shape, latent_scale
: Numeric; latent period parameters.

infectious_shape, infectious_scale
: Numeric; infectious period parameters.

resolve_shape, resolve_scale
: Numeric; resolution period parameters.

peak_day, width
: Numeric; infectivity profile.

ptrans_threshold, detect_threshold_log10, detect_threshold_Ct
: Numeric; testing thresholds.

surveillance_interval
: Integer; days between tests.

test_daily
: Logical; daily testing after detection.

viral_testing
: Character; "viral load" or "Ct".

V_ref, V_rho
: Numeric; viral load parameters.

Ct_50, Ct_delta
: Numeric; Ct parameters.

VL_params_list, Ct_params_list, household_profile_list
: Named lists.

stan_file
: Path to Stan model file.

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

stan_control
: List; Stan control list.

stan_init
: Character or function; Stan initialization.

stan_refresh
: Integer; Stan refresh.

stan_cores
: Integer; CPU cores for Stan.

Value

A list with raw_simulation, stan_data, fit, posterior_summary, and diagnostic_df.

See Also

GenSyn

postprocess_stan_fit *Tidy posterior summary for role multipliers*

Description

Extracts posterior summaries from a `stanfit`.

Usage

```
postprocess_stan_fit(fit)
```

Arguments

`fit`	A `stanfit` object.

Value

A data frame with posterior summaries.

prepare_stan_data *Prepare Stan data from diagnostic testing records*

Description

Converts raw diagnostic testing data into Stan-ready format with the `time_since_infection` matrix for soft-gate latent period modeling.

Usage

```
prepare_stan_data(
  raw_data,
  seasonal_forcing_list = NULL,
  viral_testing_mode = "viral load",
  T_max = 365,
  V_ref = 3,
  V_rho = 2.5,
  peak_day = 2,
  width = 4,
  alpha_comm_by_role = 0.005,
  reference_phi = 1,
  reference_kappa = 1
)
```

Arguments

raw_data	Data frame with columns hh_id, person_id, role, day_index, test_result, and optionally pcr_sample.
seasonal_forcing_list	Named list with seasonal forcing vectors by role.
viral_testing_mode	Character; "viral load" or "Ct".
T_max	Integer; maximum time horizon (days).
V_ref, V_rho	Numeric; viral load transmission parameters.
peak_day, width	Numeric; infectivity profile parameters.
alpha_comm_by_role	Numeric; community acquisition rate.
reference_phi, reference_kappa	Numeric; reference parameters.

Value

Named list suitable for Stan.

---

prepare_stan_households_from_user_data

Prepare per-household inputs for the Stan RSV/VL model

Description

Converts user data to a list of per-household data frames required by the Stan RSV/VL pipeline. Accepts either a long test-day table or a per-person episodes table.

Usage

```
prepare_stan_households_from_user_data(
  user_data,
  role_levels = c("adult", "child", "elderly", "toddler"),
  vl_mode = c("from_long", "auto"),
  vl_source = c("none", "column", "simulate"),
  vl_column = NULL,
  start_date = NULL,
  end_date = NULL
)
```

Arguments

user_data	Either: • *Long format:* columns HH, individual_ID, role, test_date, infection_status (optionally a VL column); or • *Per-person format:* columns hh_id, person_id, role, infection_time, infectious_start, infectious_end (optionally vl_full_trajectory).
role_levels	Character vector of allowed roles after normalization.

vl_mode	Character; currently used only to trigger trajectory building.
vl_source	Character; one of "none", "column", "simulate".
vl_column	Optional name of the VL column when vl_source = "column".
start_date, end_date	Optional Date bounds.

Value

A named list of data frames (one per household).

`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:
# Simulate and estimate
seasonal_forcing_list <- list(
  adult=rep(0.1,365), child=rep(0.1,365), elderly=rep(0.1,365), toddler=rep(0.1,365)
)
fit <- GenSyn(
  n_households=50,
  seasonal_forcing_list=seasonal_forcing_list, max_days=365,
  stan_chains=4, stan_iter=2000, stan_warmup=1000, stan_cores=4
)
print(fit)

## End(Not run)
```

print.TransmissionChainResult *Print a TransmissionChainResult*

Description

Nicely prints sections available in a `TransmissionChainResult` returned by `TransmissionChainAnalysis`.

Usage

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

Arguments

`x`	A `TransmissionChainResult` object.
`...`	Passed to or from other methods (unused).

Value

`x`, returned invisibly.

See Also

`TransmissionChainAnalysis`

Examples

```
## Not run:
T_max <- 12
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","child","elderly","adult","child","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)
)
seasonal_forcing_list <- list(
  adult = rep(1, T_max), child = rep(1, T_max),
  elderly = rep(1, T_max), toddler = rep(1, T_max)
)
result <- TransmissionChainAnalysis(
  user_data = df_person,
  seasonal_forcing_list = seasonal_forcing_list,
  max_days = T_max,
  stan_chains = 1, stan_iter = 300, stan_warmup = 150
)
print(result)

## End(Not run)
```

run_household_stan *Run the Stan household model*

Description

Wrapper around `rstan::sampling()` with reasonable defaults.

Usage

```
run_household_stan(
  stan_data,
  stan_file = "model.stan",
  chains = 4,
  iter = 2000,
  warmup = 1000,
  control = list(adapt_delta = 0.99, max_treedepth = 20),
  init = "random",
  refresh = 50,
  cores = 4,
  stan_code = NULL,
  package_name = NULL
)
```

Arguments

`stan_data`	Named list from `build_stan_household_arrays` or `prepare_stan_data`.
`stan_file`	Path to Stan model file.
`chains`, `iter`, `warmup`	Stan MCMC settings.
`control`	List passed to `rstan::sampling()`.
`init`, `refresh`, `cores`	Stan parameters.
`stan_code`	Optional character string with Stan program code.
`package_name`	Optional package name for `system.file` lookup.

Value

A `stanfit` object.

simulate_households *Simulate households (RSV/VL engine)*

Description

Generates synthetic household data for the RSV/VL–Stan pipeline. Returns a list with stacked data frame, per-household list, and long testing records (diagnostic_df).

Usage

```
simulate_households(
  n_households,
  start_date = as.Date("2024-01-01"),
  end_date = as.Date("2024-12-31"),
  max_days = 365,
  seasonal_forcing_list = NULL,
  alpha_comm_by_role = 0.005,
  beta1 = 0.3,
  beta2 = 0.05,
  delta = 0,
  phi_by_role = c(adult = 1, child = 7, toddler = 7, elderly = 4),
  kappa_by_role = c(adult = 1, child = 1.5, toddler = 1.5, elderly = 1),
  latent_shape = 3,
  latent_scale = 0.5,
  infectious_shape = 3,
  infectious_scale = 1,
  resolve_shape = 1.5,
  resolve_scale = 0.5,
  peak_day = 1,
  width = 4,
  ptrans_threshold = 0.5,
  detect_threshold_log10 = 1,
  detect_threshold_Ct = 40,
  surveillance_interval = 7,
  test_daily = FALSE,
  viral_testing = "viral load",
  V_ref = 3,
  V_rho = 2.5,
  Ct_50 = 40,
  Ct_delta = 2,
  VL_params_list = default_VL_params,
  Ct_params_list = default_Ct_params,
  household_profile_list = default_household_profile,
  verbose = FALSE
)
```

Arguments

`n_households`	Integer; number of households.
`start_date, end_date`	`Date` study window.
`max_days`	Integer; simulation horizon (days).
`seasonal_forcing_list`	Optional named list (`adult`, `child`, `elderly`, `toddler`) for seasonal forcing.
`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.

`latent_shape`, `latent_scale`
Numeric; gamma parameters for latent period.

`infectious_shape`, `infectious_scale`
Numeric; gamma parameters for infectious period.

`resolve_shape`, `resolve_scale`
Numeric; gamma parameters for resolution period.

`peak_day`, `width` Numeric; infectivity profile parameters.

`ptrans_threshold`
Numeric; transmission potential threshold.

`detect_threshold_log10`
Numeric; viral load detection threshold in 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; one of `"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 VL trajectory parameters.

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

`household_profile_list`
Named list; household composition probabilities.

`verbose` Logical; print progress information.

Value

A list with `hh_df` (stacked data frame), `households` (per-household list), and `diagnostic_df` (long testing records).

`TransmissionChainAnalysis`
TransmissionChainAnalysis: analyze user-supplied household data

---

Description

Runs the end-to-end workflow on *user-provided* data using Bayesian Stan estimation. This function does not simulate data; it expects input in tabular form (see Details).

Usage

```
TransmissionChainAnalysis(
  user_data,
  index_vl_column = "vl_test",
  start_date = as.Date("2024-01-01"),
  end_date = as.Date("2024-12-31"),
  seasonal_forcing_list = NULL,
  max_days = 365,
  stan_file = "model.stan",
  stan_chains = 4,
  stan_iter = 2000,
  stan_warmup = 1000,
  stan_control = list(adapt_delta = 0.99, max_treedepth = 20),
  stan_init = "random",
  stan_refresh = 50,
  stan_cores = 4,
  vl_mode = c("auto", "from_long"),
  vl_source = c("column", "simulate", "none"),
  vl_column = NULL,
  role_levels = c("adult", "child", "elderly", "toddler")
)
```

Arguments

`user_data`	A data.frame or a list of data.frames describing household testing or episode data (see Details for accepted shapes). Lists are row-bound internally.
`index_vl_column`	Optional character; name of the viral-load (VL) column used by SAR-by-VL plotting. Defaults to `"vl_test"` when present.
`start_date, end_date`	`Date` study window.
`seasonal_forcing_list`	Optional named list of role vectors for forcing.
`max_days`	Integer; maximum modeled days.
`stan_file`	Path to a Stan model file.
`stan_chains, stan_iter, stan_warmup`	Integers; Stan sampling controls.
`stan_control`	List; passed to `rstan::sampling(..., control = ...)`.
`stan_init`	Character or function; Stan initialization.
`stan_refresh`	Integer; Stan refresh rate.
`stan_cores`	Integer; CPU cores for Stan.
`vl_mode`	One of `c("auto","from_long")` indicating how to derive VL.
`vl_source`	One of `c("column","simulate","none")` indicating the VL source.
`vl_column`	Optional character; the VL column when `vl_source="column"`.
`role_levels`	Character vector; canonical role levels for normalization.

Details

Accepted input shapes:

- Long testing table with columns like `HH`, `individual_ID`, `role`, `test_date`, `infection_status`, and optionally a VL column.
- Per-person episode table with `hh_id`, `person_id`, `role`, `infection_time`, `infectious_start`, `infectious_end`, and optionally a viral-load trajectory.

Value

An object of class `"TransmissionChainResult"` containing:

- `$results`: Contains `fit` (the Stan object) and `posterior_summary`.
- `$postprocessing`: Stan posterior summary.
- `$plot_list`: named list of ggplot objects when built.

See Also

`main_parameter_estimation_pipeline`, `GenSyn`

Examples

```
## Not run:
T_max <- 12
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","child","elderly","adult","child","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)
)
seasonal_forcing_list <- list(
  adult = rep(1, T_max), child = rep(1, T_max),
  elderly = rep(1, T_max), toddler = rep(1, T_max)
)
result <- TransmissionChainAnalysis(
  user_data = df_person,
  seasonal_forcing_list = seasonal_forcing_list,
  max_days = T_max,
  stan_chains = 1, stan_iter = 300, stan_warmup = 150
)

## End(Not run)
```

Index