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

November 18, 2025

Title Household Transmission Chain Simulation and Estimation

Version 0.0.0.9000

Description This package provides a framework for household transmission chain analysis by a streamlined pipeline to simulate household transmission data and to estimate community and within-household infection risks from either synthetic or user-supplied data. The workflow builds person-day tables, impute infection timelines with Gamma-distributed delays, and fits penalized models with optional covariates. Post-processing summarizes mean estimates, uncertainty, bias, and relative bias, Two user functions cover the main tasks: GenSyn() to simulate data and run the full estimation pipeline (with optional summaries), and TransmissionChainAnalysis() to run the same estimation pipeline on user data.

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,
tidyverse

Suggests testthat (>= 3.0.0),
knitr,
rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

Contents

attach_vl_full_trajectory_from_long 2
build_person_day_table 3
build_stan_household_arrays 4
dataframe_to_household_list 5

data_summarization 5
generate_household_roles 6
generate_synthetic_data_one 7
GenSyn 9
g_rescaled 12
households_to_long_tests 13
infectious_time_imputation 14
main_parameter_estimation_pipeline 14
normalize_roles 18
postprocessing_estimates 18
postprocess_stan_fit 19
prepare_stan_households_from_user_data 19
print.GenSynResult 20
print.TransmissionChainResult 21
p_transmission 21
running_parameter_estimation 22
run_household_stan 23
seasonal_cosine 24
seasonal_forcing_to_series 24
simulate_households 25
simulate_multiple_households_comm 27
simulate_one_household_comm 28
simulate_viral_load_trajectory 29
summarize_individuals 30
TransmissionChainAnalysis 31

Index **34**

`attach_vl_full_trajectory_from_long` *Attach viral-load trajectories to person-level data*

Description

Creates or fills a `vl_full_trajectory` list-column for each person over their infectious span using VL from a long table, simulated VL, or a sentinel.

Usage

```
attach_vl_full_trajectory_from_long(
  hh_long,
  hh_person,
  user_data,
  vl_source = c("column", "simulate", "none"),
  vl_column = NULL,
  start_date,
  end_date
)
```

Arguments

hh_long	Long test-day data (may be NULL if simulating); must include HH, individual_ID, test_date and the VL column when vl_source = "column".
hh_person	Person-level data frame with infection/imputation columns.
user_data	Original user input (not modified; passed for convenience).
vl_source	One of "column", "simulate", "none".
vl_column	Column name with log10 VL when vl_source = "column".
start_date, end_date	Study window (Date); used for alignment.

Value

hh_person with a vl_full_trajectory list-column added.

---

build_person_day_table *Construct person-day long data*

Description

Expands individual timelines into daily rows and computes within-household infectious counts by infector role for likelihood-based estimation.

Usage

```
build_person_day_table(dt, tmax, cases_t, covariate_cols = character(0))
```

Arguments

dt	data.table from infectious_time_imputation.
tmax	Integer; maximum day index.
cases_t	Numeric of length tmax + 1; community intensity for days 0..tmax.
covariate_cols	Character; names of covariates to carry (scalars or day series).

Value

data.table with columns: agegrp2, agegrp3, agegrp4, n_inf, n_inf_infant, n_inf_sibling, n_inf_adult, n_inf_elder, cases, event, ID_indiv, ID_hh, day, and requested covariates.

build_stan_household_arrays *Assemble Stan data arrays for the household model*

Description

Produces the data list for Stan from simulator households or a person-level frame, including infectious indicators, infection days, VL arrays, household membership, and seasonal forcing.

Usage

```
build_stan_household_arrays(
  households,
  T_max = 365,
  seasonal_forcing_list,
  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 = 2
)
```

Arguments

households	Either a list of per-household data frames (each with vl_full_trajectory) or a single person-level data frame with hh_id, role, infection_time, infectious_start, infectious_end, infection_resolved, vl_full_trajectory.
T_max	Integer; time horizon (days).
seasonal_forcing_list	Named list of role-specific forcing series.
alpha_comm_by_role, beta1, beta2, V_ref, reference_phi, reference_kappa	Scalars controlling hazards and reference levels.
g_peak_day, g_width	Scalars controlling the infectivity profile.

Details

If VL is absent or sentinel-only, V_term is zero and the VL-mediated component is effectively disabled.

Value

A named list suitable for data= in Stan containing N, T, H, R, delta, hh_id, role_id, Y, I, V, V_term, alpha_comm_by_role, max_infectious, hh_size, hh_max_size, hh_members, g_profile, V_ref, beta1, beta2, reference_phi, reference_kappa, seasonal_forcing_mat.

dataframe_to_household_list

Convert a user data frame to a list of household tables

Description

Splits a long-format test table into a list of per-household `data.tables`, preserving the required columns and (optionally) any extra covariates.

Usage

```
dataframe_to_household_list(
  df,
  hh_col = "HH",
  id_col = "individual_ID",
  role_col = "role",
  date_col = "test_date",
  inf_col = "infection_status",
  comm_col = "community_risk",
  keep_extra_cols = TRUE
)
```

Arguments

`df`	Data frame with at least household ID, individual ID, role, test date, infection status, and community risk columns.
`hh_col`, `id_col`, `role_col`, `date_col`, `inf_col`, `comm_col`	Character. Column names for household ID (default `"HH"`), individual ID (`"individual_ID"`), role (`"role"`), test date (`"test_date"`), infection status (`"infection_status"`), and community risk (`"community_risk"`).
`keep_extra_cols`	Logical; keep additional user columns (default `TRUE`).

Value

List of `data.tables`, one per household (household ID placed first).

data_summarization

Summarize infection episodes (one row per individual)

Description

Collapses long testing records to episode-level features and optional covariate summaries. Supply either a single data frame `df` with the six core columns, or provide all six vectors individually. An optional covariate frame can be merged.

Usage

```
data_summarization(
  df = NULL,
  Household_ID = NULL,
  Individual_ID = NULL,
  Household_role = NULL,
  Sample_test_days = NULL,
  Infectious_status = NULL,
  Community_rate_infection = NULL,
  Covariate_DataFrame = NULL,
  covariate_cols = NULL
)
```

Arguments

`df`	Data frame with columns `HH`, `individual_ID`, `role`, `test_date`, `infection_status`, `community_risk`. Ignored when all six vector inputs are provided.
`Household_ID`, `Individual_ID`, `Household_role`, `Sample_test_days`, `Infectious_status`, `Community_rate_infection`	Vectors used only when `df` is `NULL`.
`Covariate_DataFrame`	Optional data frame to merge before summarizing; joined by available keys among `c("HH","individual_ID","test_date")` (at least `individual_ID` required).
`covariate_cols`	Optional character vector restricting which covariates from `Covariate_DataFrame` are summarized.

Details

Records are ordered by `HH`, `individual_ID`, `test_date`. Episodes are runs of `infection_status == 1`; first-episode fields refer to the earliest run.

Value

A data frame with one row per individual including: `HH`, `individual_ID`, episode counts and timing fields (e.g., `infection.detected.start`, `infection.detected.end`, `infection.true.duration`, `last_negative`), `infection.infectious.day` (comma-separated days), `community.risk`, `role`, and (if present) basic summaries for covariates.

See Also

`summarize_individuals`, `main_parameter_estimation_pipeline`

`generate_household_roles` *Generate a simple household role composition*

Description

Samples a household (size 3–5) with at least one child and two adults, optionally adding elderly/toddler roles.

Usage

```
generate_household_roles()
```

Value

Character vector of role labels.

---

generate_synthetic_data_one *Generate standardized synthetic data for one household*

Description

Simulates test-day observations for one household over a date window with community seasonality, within-household transmission, adaptive testing, baseline/partial immunity, and optional covariates.

Usage

```
generate_synthetic_data_one(
  household_id,
  hh.size = sample(3:7, 1),
  tests.per.week = 2,
  p.comm.base.infant.fix = 0.001,
  p.comm.multiplier.sibling = 1,
  p.comm.multiplier.parent = 1,
  p.comm.multiplier.elder = 1,
  p.hh.base.infant = 0.1,
  p.hh.multiplier.sibling = 1,
  p.hh.multiplier.parent = 1,
  p.hh.multiplier.elder = 1,
  p.imm.base.sibling = 1e-10,
  p.imm.base.parent = 1e-10,
  p.imm.base.elder = 1e-10,
  partial.immunity.infant = 1e-10,
  partial.immunity.sibling = 1e-10,
  partial.immunity.parent = 1e-10,
  partial.immunity.elder = 1e-10,
  duration.latent = 2,
  duration.infect.inf = 3,
  multiplier.dur.sibpar = 0.5,
  p.detect = 0.999,
  amplitude = 0,
  phase = -0.408,
  start_date = as.Date("2024-09-21"),
  end_date = as.Date("2025-04-17"),
  Covariates = FALSE,
  Covariates_list = c("Vaccination status", "Antibody Level"),
  Covariate_specifications = NULL
)
```

Arguments

household_id	Integer; household identifier written to HH.
hh.size	Integer; household size.
tests.per.week	Integer; tests per person per week.
p.comm.base.infant.fix	Numeric; baseline community infection prob/day (infant).
p.comm.multiplier.sibling, p.comm.multiplier.parent, p.comm.multiplier.elder	Numeric; community multipliers by role.
p.hh.base.infant	Numeric; baseline within-household infection prob/day (infant source).
p.hh.multiplier.sibling, p.hh.multiplier.parent, p.hh.multiplier.elder	Numeric; within-household multipliers by source role.
p.imm.base.sibling, p.imm.base.parent, p.imm.base.elder	Numeric; baseline immunity at day 1.
partial.immunity.infant, partial.immunity.sibling, partial.immunity.parent, partial.immunity.elder	Numeric; partial-immunity modifiers by role.
duration.latent	Integer; mean latent period (days).
duration.infect.inf	Integer; mean infectious duration for infants (days).
multiplier.dur.sibpar	Numeric; infectious-duration multiplier for non-infants.
p.detect	Numeric; detection probability if infected on a test day.
amplitude, phase	Numeric; seasonality parameters for community risk.
start_date, end_date	Date; simulation window.
Covariates	Logical; generate additional covariates.
Covariates_list	Character; covariate names.
Covariate_specifications	List; optional per-covariate specs.

Value

Data frame with columns HH, individual_ID, role, test_date (1 = start_date), infection_status, community_risk, and optional covariates.

GenSyn *GenSyn: household simulation & estimation wrapper*

Description

Runs `main_parameter_estimation_pipeline()` to either simulate (or ingest) household data, fit via the legacy MLE path or the RSV/VL+Stan path, and assemble a `"GenSynResult"`.

Usage

```
GenSyn(
  n_households = 10,
  n_runs = 10,
  data_summary = FALSE,
  print_plots = TRUE,
  plots = c("daily", "weekly", "timeline", "sar"),
  engine = c("legacy", "rsv_vl"),
  estimation_method = c("mle", "stan"),
  index_vl_column = "vl_test",
  user_data = NULL,
  synthetic_data = TRUE,
  hh.size = sample(3:7, 1),
  tests.per.week = 1,
  Covariates = FALSE,
  Covariates_list = c("Vaccination status", "Antibody Level"),
  Covariate_specifications = NULL,
  day_series_covariates = TRUE,
  series_cols = NULL,
  comm_covariate_cols = NULL,
  hh_covariate_cols = NULL,
  hh_by_role = FALSE,
  hh_role_covariate_cols = NULL,
  standardize_covariates = TRUE,
  lambda_comm = 0.01,
  lambda_hh = 0.01,
  p.comm.base.infant.fix = 0.002,
  p.comm.multiplier.sibling = 1,
  p.comm.multiplier.parent = 1,
  p.comm.multiplier.elder = 1,
  p.hh.base.infant = 0.2,
  p.hh.multiplier.sibling = 0.5267686,
  p.hh.multiplier.parent = 0.8008933,
  p.hh.multiplier.elder = 0.6008933,
  p.imm.base.sibling = 1e-10,
  p.imm.base.parent = 1e-10,
  p.imm.base.elder = 1e-10,
  partial.immunity.infant = 1e-10,
  partial.immunity.sibling = 1e-10,
  partial.immunity.parent = 1e-10,
  partial.immunity.elder = 1e-10,
  duration.latent = 1,
```

```
  duration.infect.inf = 2,
  multiplier.dur.sibpar = 0.5,
  p.detect = 0.999,
  amplitude = 2.6581 * 0,
  phase = -0.408,
  start_date = as.Date("2024-09-21"),
  end_date = as.Date("2025-04-17"),
  latent_par = list(shape = 2, scale = 1),
  report_par = list(shape = 1, scale = 1.5),
  infect_par = list(shape = 3, scale = 2),
  start_par = c(-6, 0.02, -2, rep(0, 6)),
  lambda = 0.01,
  lambda0 = 0.2,
  lambda_alpha = 5,
  delta0_true = qlogis(0.002),
  alpha0_true = qlogis(0.2),
  seasonal_forcing_list = NULL,
  max_days = 365,
  stan_file = "../inst/stan/HH_parameter_estimation2.stan",
  stan_chains = 1,
  stan_iter = 100,
  stan_warmup = 100,
  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 when `synthetic_data=TRUE`.
`n_runs`	Integer; number of repeated MLE runs (legacy path only).
`data_summary`	Logical (legacy+MLE only); if `TRUE`, per-individual summary is retained/printed.
`print_plots`	Logical; print plots if produced.
`plots`	Character vector of plot names (`"daily"`, `"weekly"`, `"sar"`, `"timeline"`) or `"all"`.
`engine`	Character; one of `c("legacy","rsv_vl")`.
`estimation_method`	Character; one of `c("mle","stan")`.
`index_vl_column`	Character; viral-load column name for plotting on Stan path (default `"vl_test"`).
`user_data`	Optional user data (see `main_parameter_estimation_pipeline`); ignored if `synthetic_data=TRUE`.
`synthetic_data`	Logical; simulate data internally.
`hh.size`	Integer; household size used in simulation.
`tests.per.week`	Integer; tests per person per week (legacy engine).
`Covariates`	Logical; include synthetic covariates (legacy).
`Covariates_list`	Character vector; names of covariates (legacy).

`Covariate_specifications`
: Optional list describing covariate generation (legacy).

`day_series_covariates`
: Logical; include day-series covariates (legacy+MLE).

`series_cols` Optional character vector; selected day-series columns (legacy+MLE).

`comm_covariate_cols`, `hh_covariate_cols`
: Optional character vectors; likelihood-level covariates (legacy+MLE).

`hh_by_role` Logical; fit separate HH effects by role (legacy+MLE).

`hh_role_covariate_cols`
: Optional named list of role-specific covariates (legacy+MLE).

`standardize_covariates`
: Logical; standardize covariates (legacy+MLE).

`lambda_comm`, `lambda_hh`
: Numeric; ridge penalties for community/household covariates (legacy+MLE).

`p.comm.base.infant.fix`, `p.comm.multiplier.sibling`, `p.comm.multiplier.parent`, `p.comm.multiplier.elder`
: Numeric; community risk knobs (legacy simulator).

`p.hh.base.infant`, `p.hh.multiplier.sibling`, `p.hh.multiplier.parent`, `p.hh.multiplier.elder`
: Numeric; within-HH knobs (legacy simulator).

`p.imm.base.sibling`, `p.imm.base.parent`, `p.imm.base.elder`, `partial.immunity.infant`, `partial.immunity.sibling`, `partial.immunity.parent`, `partial.immunity.elder`
: Numeric; immunity knobs (legacy simulator).

`duration.latent`, `duration.infect.inf`, `multiplier.dur.sibpar`
: Numeric; timing knobs (legacy simulator).

`p.detect` Numeric; detection probability (legacy simulator).

`amplitude`, `phase`
: Numeric; seasonality controls (legacy simulator).

`start_date`, `end_date`
: `Date`; study window.

`latent_par`, `report_par`, `infect_par`
: Lists of gamma parameters (shape, scale) for imputation (legacy+MLE).

`start_par` Numeric vector; optimizer starting values (legacy+MLE).

`lambda`, `lambda0`, `lambda_alpha`
: Numeric; optimizer penalties (legacy+MLE).

`delta0_true`, `alpha0_true`
: Numeric; reference values on logit scale (legacy+MLE).

`seasonal_forcing_list`
: Optional named list of role vectors; seasonal forcing (RSV/VL+Stan).

`max_days` Integer; maximum simulated days (RSV/VL+Stan).

`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.

Details

Arguments are forwarded to main_parameter_estimation_pipeline(); some are ignored depending on engine/estimation_method.

Value

A "GenSynResult" list with elements: $call, $engine, $estimation_method, $n_households, $n_runs, $results, $postprocessing, $plot_list.

See Also

main_parameter_estimation_pipeline, postprocessing_estimates

Examples

```
# Lightweight MLE example (no Stan):
set.seed(1)
fit <- GenSyn(
  n_households = 3,
  engine = "legacy", estimation_method = "mle",
  n_runs = 2, print_plots = FALSE
)
class(fit)

## Not run:
# Heavier Stan example (do not run on CRAN):
seasonal_forcing_list <- list(
  adult=rep(1,60), child=rep(1,60), elderly=rep(1,60), toddler=rep(1,60)
)
fit2 <- GenSyn(
  n_households=2,
  engine="rsv_vl", estimation_method="stan",
  seasonal_forcing_list=seasonal_forcing_list, max_days=60,
  stan_chains=1, stan_iter=200, stan_warmup=100, stan_cores=1
)

## End(Not run)
```

---

g_rescaled *Gaussian-like infectivity profile (normalized)*

Description

Bell-shaped infectivity profile scaled to have maximum 1.

Usage

```
g_rescaled(t, peak_day, width)
```

Arguments

t	Integer vector of times (e.g., days since infectious onset).
peak_day	Integer; peak day.
width	Numeric; spread around the peak.

Value

Numeric vector in [0,1] of length `t`.

`households_to_long_tests` *Flatten simulator households to a long test-day table*

Description

Expands simulator output to per-person per-day rows suitable for legacy processing and plotting.

Flattens simulator output (list of per-household data frames) into a long table with one row per person-day.

Usage

```
households_to_long_tests(households, seasonal_forcing_list = NULL)

households_to_long_tests(households, seasonal_forcing_list = NULL)
```

Arguments

`households`	List of household data frames from the simulator (each may carry attributes such as `test_days` and per-person `viral_loads_test_days`).
`seasonal_forcing_list`	Optional named list of role-specific forcing series (`adult`, `child`, `elderly`, `toddler`).

Details

If `test_days` is absent, the horizon is inferred from `infectious_end`/`infection_resolved` (fallback to 1). Missing `infection_status` is treated as 0. Household index in the list is used for `HH`.

Value

Data frame with columns: `HH`, `individual_ID`, `role`, `test_date`, `infection_status` and (if available) `community_risk`.

A data frame with columns `HH`, `individual_ID`, `role`, `test_date`, `infection_status`, `community_risk`, `vl_test`.

infectious_time_imputation

Impute infection timelines from delay distributions

Description

Imputes infection date, infectious start/end, and component delays using gamma distributions, optionally scaled by covariate functions.

Usage

```
infectious_time_imputation(
  dt,
  study_start,
  latent_par,
  report_par,
  infect_par,
  latent_scale_fn = NULL,
  report_scale_fn = NULL,
  infect_scale_fn = NULL
)
```

Arguments

dt	data.table from summarize_individuals.
study_start	Date origin for relative day indices.
latent_par, report_par, infect_par	Lists with shape and scale for latent, reporting, and infectious periods.
latent_scale_fn, report_scale_fn, infect_scale_fn	Optional functions taking dt[idx] (infected rows) and returning numeric scale multipliers.

Value

The input dt with columns latent_delay, report_delay, infect_period, inf_date, inf_start_date, inf_end_date, and relative-day variants.

main_parameter_estimation_pipeline

Main parameter estimation pipeline (simulate or analyze)

Description

End-to-end workflow for household transmission estimation via legacy MLE or RSV/VL+Stan. Works with simulated or user-provided data.

Usage

```
main_parameter_estimation_pipeline(
  user_data = NULL,
  synthetic_data = TRUE,
  engine = c("legacy", "rsv_vl"),
  estimation_method = c("mle", "stan"),
  n_households = 10,
  n_runs = 10,
  hh.size = sample(3:7, 1),
  tests.per.week = 1,
  Covariates = FALSE,
  Covariates_list = c("Vaccination status", "Antibody Level"),
  Covariate_specifications = NULL,
  day_series_covariates = TRUE,
  series_cols = NULL,
  comm_covariate_cols = NULL,
  hh_covariate_cols = NULL,
  hh_by_role = FALSE,
  hh_role_covariate_cols = NULL,
  standardize_covariates = TRUE,
  lambda_comm = 0.01,
  lambda_hh = 0.01,
  p.comm.base.infant.fix = 0.002,
  p.comm.multiplier.sibling = 1,
  p.comm.multiplier.parent = 1,
  p.comm.multiplier.elder = 1,
  p.hh.base.infant = 0.2,
  p.hh.multiplier.sibling = 0.5267686,
  p.hh.multiplier.parent = 0.8008933,
  p.hh.multiplier.elder = 0.6008933,
  p.imm.base.sibling = 1e-10,
  p.imm.base.parent = 1e-10,
  p.imm.base.elder = 1e-10,
  partial.immunity.infant = 1e-10,
  partial.immunity.sibling = 1e-10,
  partial.immunity.parent = 1e-10,
  partial.immunity.elder = 1e-10,
  duration.latent = 1,
  duration.infect.inf = 2,
  multiplier.dur.sibpar = 0.5,
  p.detect = 0.999,
  amplitude = 2.6581 * 0,
  phase = -0.408,
  start_date = as.Date("2024-09-21"),
  end_date = as.Date("2025-04-17"),
  latent_par = list(shape = 2, scale = 1),
  report_par = list(shape = 1, scale = 1.5),
  infect_par = list(shape = 3, scale = 2),
  start_par = c(-6, 0.02, -2, rep(0, 6)),
  lambda = 0.01,
  lambda0 = 0.2,
  lambda_alpha = 5,
```

```
  delta0_true = qlogis(0.002),
  alpha0_true = qlogis(0.2),
  seasonal_forcing_list = NULL,
  max_days = 365,
  stan_file = "HH_parameter_estimation2.stan",
  stan_chains = 1,
  stan_iter = 100,
  stan_warmup = 100,
  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 as described in `TransmissionChainAnalysis`.
`synthetic_data`	Logical; if `TRUE`, simulate data internally.
`engine`	Character; `"legacy"` or `"rsv_vl"`.
`estimation_method`	Character; `"mle"` or `"stan"` (valid pairs: `legacy+mle`, `rsv_vl+stan`).
`n_households`, `n_runs`	Integers; simulation size and number of repeated runs (MLE only).
`hh.size`, `tests.per.week`	Integers; simulation controls (legacy).
`Covariates`	Logical; include synthetic covariates (legacy).
`Covariates_list`	Character vector; covariate names (legacy).
`Covariate_specifications`	Optional list; covariate generation controls (legacy).
`day_series_covariates`	Logical; include day-series covariates (legacy+MLE).
`series_cols`	Optional character vector; day-series columns (legacy+MLE).
`comm_covariate_cols`, `hh_covariate_cols`	Optional character vectors; likelihood-level covariates (legacy+MLE).
`hh_by_role`	Logical; fit separate HH effects by role (legacy+MLE).
`hh_role_covariate_cols`	Optional named list; role-specific covariates (legacy+MLE).
`standardize_covariates`	Logical; standardize covariates (legacy+MLE).
`lambda_comm`, `lambda_hh`	Numeric; ridge penalties (legacy+MLE).
`p.comm.base.infant.fix`, `p.comm.multiplier.sibling`, `p.comm.multiplier.parent`, `p.comm.multiplier.elder`	Numeric; community knobs (legacy).
`p.hh.base.infant`, `p.hh.multiplier.sibling`, `p.hh.multiplier.parent`, `p.hh.multiplier.elder`	Numeric; within-HH knobs (legacy).

`p.imm.base.sibling`, `p.imm.base.parent`, `p.imm.base.elder`, `partial.immunity.infant`, `partial.immunity.sibling`, `partial.immunity.parent`, `partial.immunity.elder`
: Numeric; immunity knobs (legacy).

`duration.latent`, `duration.infect.inf`, `multiplier.dur.sibpar`
: Numeric; timing knobs (legacy).

`p.detect`
: Numeric; detection probability (legacy).

`amplitude`, `phase`
: Numeric; seasonality controls (legacy).

`start_date`, `end_date`
: `Date`; analysis window.

`latent_par`, `report_par`, `infect_par`
: Lists (shape, scale) for imputation (legacy+MLE).

`start_par`
: Numeric; optimizer start (legacy+MLE).

`lambda`, `lambda0`, `lambda_alpha`
: Numeric; optimizer penalties (legacy+MLE).

`delta0_true`, `alpha0_true`
: Numeric; reference values on logit scale (legacy+MLE).

`seasonal_forcing_list`
: Optional role-named list for seasonality (RSV/VL+Stan).

`max_days`
: Integer; maximum days (RSV/VL+Stan).

`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.

`stan_cores`
: Integer; CPU cores for Stan.

Details

Enforces valid engine/estimation pairs; other combinations error.

Value

For legacy+MLE: list with `raw_simulation`, `summarized_data`, `person_day`, `estimates`, and `postprocessing`. For RSV/VL+Stan: list with `raw_simulation`, `stan_data`, `fit`, `posterior_summary` (also in `postprocessing`), and `NULL` placeholders for `summarized_data`/`person_day`.

See Also

`TransmissionChainAnalysis`, `postprocessing_estimates`, `build_stan_household_arrays`, `run_household_stan`

normalize_roles *Normalize role labels to child, toddler, adult, elderly*

Description

Maps common synonyms (`infant->toddler`, `sibling->child`, `parent->adult`, `elder->elderly`) and returns a factor with consistent levels.

Usage

```
normalize_roles(x)
```

Arguments

`x`	Character vector of role labels.

Value

A factor with levels `c("child","toddler","adult","elderly")`.

postprocessing_estimates *Summarize and compare estimates against reference values*

Description

Computes mean, SD, SE across runs and (optionally) bias and relative bias versus supplied reference (true) values for each parameter.

Usage

```
postprocessing_estimates(theta_mat, true_values = NULL)
```

Arguments

`theta_mat`	Numeric matrix. Parameter estimates from multiple runs (e.g., output of `running_parameter_estim` or a Stan summary). Must be non-empty.
`true_values`	Optional named numeric vector of reference values. Names should match columns in `theta_mat`. Unmatched names are ignored; missing references yield `NA` bias/relative bias.

Details

Rows with incomplete estimates are dropped using the first column as a sentinel. If `theta_mat` lacks column names, generic names `"par1"`, `"par2"`, ... are assigned. The number of retained runs is stored in the `"n_runs"` attribute of the returned table.

This helper understands both the legacy parameter naming (e.g., `delta0`, `alpha0`, `gamma2`, `beta2`, `theta_comm_*`, `theta_hh_*`) and the new RSV/seasonality framework parameters (e.g., `phi_*`, `kappa_*`).

Value

A data.table with one row per parameter and columns:

- Parameter: column name from theta_mat
- Estimate: mean across runs
- SD: standard deviation across runs
- SE: standard error (SD / sqrt(n_runs))
- True: supplied reference value (if any)
- Bias: Estimate - True
- RelBias: Bias / |True| (NA when True is 0 or missing)
- Block, Role: simple parameter grouping labels for readability

---

postprocess_stan_fit *Tidy posterior summary for role multipliers*

Description

Extracts posterior summaries for phi_by_role[] and kappa_by_role[] from a stanfit.

Usage

```
postprocess_stan_fit(fit)
```

Arguments

fit	A stanfit object returned by run_household_stan.

Value

A data frame with columns: Parameter, Estimate, SD, Lower CI, Median, Upper CI.

---

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, e.g., c("adult","child","elderly","toddl
vl_mode	Character; currently used only to trigger trajectory building from long data ("from_long").
vl_source	Character; one of "none", "column", "simulate" (applies to long-format input).
vl_column	Optional name of the VL column when vl_source = "column".
start_date, end_date	Optional Date bounds to convert test_date to day indices (long format).

Value

A named list of data frames (one per household) with columns hh_id, person_id, role, infection_time, infectious_start, infectious_end, detection_time, infection_resolved, and vl_full_trajectory (list-column).

---

print.GenSynResult *Print method for* GenSynResult

Description

Formats a GenSynResult for display. Prints engine/method, basic run info, any stored post-processing table, and (legacy+MLE only) a short individual-level preview when x$data_summary is TRUE.

Usage

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

Arguments

x	A GenSynResult object (from GenSyn).
...	Unused.

Value

x, invisibly.

See Also

GenSyn

print.TransmissionChainResult *Print a TransmissionChainResult*

Description

Nicely prints sections available in a `TransmissionChainResult` returned by `TransmissionChainAnalysis`. For MLE, shows the post-processing table (if present) and, optionally, a preview of the individual-level summary. For Stan, shows the Stan posterior summary (or a preview) when available.

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`

p_transmission *Transmission potential as a function of viral load*

Description

Computes a saturating transmission potential on [0,1] from log10 viral load.

Usage

```
p_transmission(VL, n, p_v, K, Vm, sai)
```

Arguments

`VL`	Numeric vector; log10 viral load.
`n, p_v, K, Vm, sai`	Numeric scalars; model constants.

Value

Numeric vector of transmission potentials.

`running_parameter_estimation` *Run penalized maximum likelihood parameter estimation*

Description

Repeats optimization `n_runs` times from jittered starts to estimate community and household transmission parameters with optional covariate effects and quadratic penalties.

Usage

```
running_parameter_estimation(
  long_dt,
  n_runs,
  start_par = NULL,
  lambda,
  lambda0,
  lambda_alpha,
  delta0_true,
  alpha0_true,
  comm_covariate_cols = NULL,
  hh_covariate_cols = NULL,
  hh_by_role = FALSE,
  hh_role_covariate_cols = NULL,
  standardize_covariates = FALSE,
  lambda_comm = lambda,
  lambda_hh = lambda,
  verbose = TRUE
)
```

Arguments

`long_dt`	data.table or data.frame. Person day data from `build_person_day_table`.
`n_runs`	Integer. Number of optimization runs for multi start.
`start_par`	Numeric vector or `NULL`. Initial parameters. If `NULL`, a vector of the correct length is initialized with `delta0_true` and `alpha0_true` in the intercept slots and zeros elsewhere.
`lambda`	Numeric. Base L2 penalty on `gamma` age effects and `z_*` role offsets.
`lambda0, lambda_alpha`	Numeric. Penalty strengths that pull `delta0` and `alpha0` toward `delta0_true` and `alpha0_true`.
`delta0_true, alpha0_true`	Numeric anchors for the intercept penalties on the logit scale.
`comm_covariate_cols`	Character vector of community risk covariate names with no intercept.
`hh_covariate_cols`	Character vector of household covariates shared across roles.
`hh_by_role`	Logical. If `TRUE`, allow role specific household covariates through `hh_role_covariate_cols`.

hh_role_covariate_cols
: Named list with elements infant, sibling, adult, elder that give covariate names per role. Falls back to hh_covariate_cols if a role list is missing.

standardize_covariates
: Logical. Z score non binary numeric columns in model matrices.

lambda_comm, lambda_hh
: Numeric. L2 penalties for community and household covariate coefficients.

verbose
: Logical. If TRUE, print notes on dropped or unknown covariates and initialization.

Value

A numeric matrix of dimension n_runs by n_parameters with column names that match the parameter layout, for example delta0, gamma2, alpha0, z_sib, and theta_*.

---

run_household_stan *Run the Stan household model*

Description

Thin wrapper around rstan::stan() (or compiled model via rstan::stan_model()) with reasonable defaults.

Usage

```
run_household_stan(
  stan_data,
  stan_file = "HH_parameter_estimation2.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 as returned by build_stan_household_arrays.

stan_file
: Path to a Stan model file (or "builtin:<key>").

chains, iter, warmup
: Stan MCMC settings.

control
: List passed to rstan::stan() (e.g., adapt_delta, max_treedepth).

init, refresh, cores
: Passed through to rstan::sampling().

stan_code
: Optional character string with Stan program code.

package_name
: Optional package name for system.file("stan", ...) lookup.

Value

A `stanfit` object.

---

seasonal_cosine *Cosine seasonal multiplier (non-negative)*

Description

Cosine-based seasonal multiplier bounded below by 0.

Usage

```
seasonal_cosine(day, peak_day = 350, amplitude = 0.8, period = 365)
```

Arguments

`day`	Integer vector of days.
`peak_day`	Integer; day of maximum.
`amplitude`	Numeric; cosine amplitude.
`period`	Integer; period length.

Value

Numeric vector of multipliers.

---

seasonal_forcing_to_series *Build a role-specific seasonal forcing matrix*

Description

Stacks role-specific forcing series into a $T \times 4$ matrix for Stan.

Usage

```
seasonal_forcing_to_series(seasonal_forcing_list, T_max = 365)
```

Arguments

`seasonal_forcing_list`	Named list with entries `adult`, `child`, `elderly`, `toddler`; each a numeric vector of length $\geq$ `T_max`.
`T_max`	Integer; number of rows (days).

Value

A numeric matrix with columns `adult`, `child`, `elderly`, `toddler` and `T_max` rows.

`simulate_households` *Simulate households (legacy or RSV/VL engine)*

Description

Generates synthetic household data for downstream MLE or RSV/VL–Stan pipelines. Returns per-household data frames (legacy) or a long test-day table with an attached per-household list (RSV/VL).

Usage

```
simulate_households(
  n_households,
  hh.size,
  tests.per.week = 1,
  engine = c("legacy", "rsv_vl"),
  simulation_function = generate_synthetic_data_one,
  Covariates = FALSE,
  Covariates_list = c("Vaccination status", "Antibody Level"),
  Covariate_specifications = NULL,
  amplitude = 0,
  phase = 0,
  start_date = as.Date("2024-09-21"),
  end_date = as.Date("2025-04-17"),
  max_days = NULL,
  p.comm.base.infant.fix = 0.002,
  p.comm.multiplier.sibling = 1,
  p.comm.multiplier.parent = 1,
  p.comm.multiplier.elder = 1,
  p.hh.base.infant = 0.2,
  p.hh.multiplier.sibling = 0.5267686,
  p.hh.multiplier.parent = 0.8008933,
  p.hh.multiplier.elder = 0.6008933,
  p.imm.base.sibling = 1e-10,
  p.imm.base.parent = 1e-10,
  p.imm.base.elder = 1e-10,
  partial.immunity.infant = 1e-10,
  partial.immunity.sibling = 1e-10,
  partial.immunity.parent = 1e-10,
  partial.immunity.elder = 1e-10,
  duration.latent = 1,
  duration.infect.inf = 2,
  multiplier.dur.sibpar = 0.5,
  p.detect = 0.999,
  seasonal_forcing_list = NULL
)
```

Arguments

`n_households`	Integer; number of households.
`hh.size`	Integer or vector; household size(s).

`tests.per.week` Integer; testing frequency (legacy).

`engine` Character; one of `c("legacy","rsv_vl")`.

`simulation_function`
Function used by the legacy engine to simulate one household (defaults to `generate_synthetic_dat`

`Covariates` Logical; include synthetic covariates (legacy).

`Covariates_list`
Character vector; names of covariates (legacy).

`Covariate_specifications`
Optional list; covariate generation controls (legacy).

`amplitude`, `phase`
Numeric seasonality knobs (legacy/upstream only).

`start_date`, `end_date`
`Date` study window (RSV/VL path and inference).

`max_days` Optional integer; horizon to stamp as attribute `"test_days"` on each household.

`p.comm.base.infant.fix`
Numeric; base community risk (infant; legacy).

`p.comm.multiplier.sibling`
Numeric; community multiplier (sibling; legacy).

`p.comm.multiplier.parent`
Numeric; community multiplier (parent; legacy).

`p.comm.multiplier.elder`
Numeric; community multiplier (elder; legacy).

`p.hh.base.infant`
Numeric; base HH transmission (infant; legacy).

`p.hh.multiplier.sibling`
Numeric; HH multiplier (sibling; legacy).

`p.hh.multiplier.parent`
Numeric; HH multiplier (parent; legacy).

`p.hh.multiplier.elder`
Numeric; HH multiplier (elder; legacy).

`p.imm.base.sibling`
Numeric; baseline immunity (sibling; legacy).

`p.imm.base.parent`
Numeric; baseline immunity (parent; legacy).

`p.imm.base.elder`
Numeric; baseline immunity (elder; legacy).

`partial.immunity.infant`
Numeric; partial immunity (infant; legacy).

`partial.immunity.sibling`
Numeric; partial immunity (sibling; legacy).

`partial.immunity.parent`
Numeric; partial immunity (parent; legacy).

`partial.immunity.elder`
Numeric; partial immunity (elder; legacy).

`duration.latent`
Integer; latent duration (legacy).

`duration.infect.inf`
: Integer; infectious duration (legacy).

`multiplier.dur.sibpar`
: Numeric; duration multiplier for sibling/parent (legacy).

`p.detect`
: Numeric; detection probability per test (legacy).

`seasonal_forcing_list`
: Optional named list (`adult`, `child`, `elderly`, `toddler`) for RSV/VL downstream use.

Value

If `engine="legacy"`: a list of per-household data frames (each may carry `attr(., "test_days")`). If `engine="rsv_vl"`: a long test-day data frame with attribute `"households"` holding the per-household list.

`simulate_multiple_households_comm` *Simulate multiple households (RSV/VL engine)*

Description

Runs `simulate_one_household_comm` across households and returns both the stacked data frame and the per-household list.

Usage

```
simulate_multiple_households_comm(
  n_households = 200,
  alpha_comm_by_role = 0.005,
  beta1 = 0.06,
  beta2 = 1e-07,
  delta = 1,
  rho = 3,
  V_ref = 1e+07,
  phi_by_role = c(adult = 1, child = 1.1, elderly = 0.8, toddler = 1.2),
  kappa_by_role = c(adult = 1, child = 1.1, elderly = 0.7, toddler = 1.2),
  latent_shape = 2,
  latent_scale = 1,
  infectious_shape = 2,
  infectious_scale = 2,
  peak_day = 2,
  width = 2,
  max_days = 40,
  verbose = FALSE,
  seasonal_forcing_list = NULL
)
```

Arguments

`n_households`	Integer; number of households.
`alpha_comm_by_role`	Numeric; baseline community hazard (scalar).
`beta1`, `beta2`, `delta`, `rho`, `V_ref`	Numeric; transmission constants.
`phi_by_role`, `kappa_by_role`	Named numeric vectors; susceptibility and infectivity multipliers by role.
`latent_shape`, `latent_scale`, `infectious_shape`, `infectious_scale`	Numeric; gamma parameters for latent/infectious durations.
`peak_day`, `width`	Numeric; infectivity profile controls.
`max_days`	Integer; simulation horizon (days).
`verbose`	Logical; print progress.
`seasonal_forcing_list`	Optional list of length 4 with role-specific seasonal multipliers (length `max_days`).

Value

A list with:

`hh_df` Stacked data frame across households.

`households` List of per-household data frames.

`simulate_one_household_comm` *Simulate one household (RSV/VL engine)*

Description

Simulates infection events, infectious periods, detection, and VL trajectories for a single household over a fixed horizon.

Usage

```
simulate_one_household_comm(
  hh_id,
  roles,
  alpha_comm_by_role,
  beta1 = 0.06,
  beta2 = 1e-07,
  delta = 1,
  rho = 3,
  V_ref = 1e+07,
  phi_by_role = c(adult = 1, child = 1.1, elderly = 0.8, toddler = 1.2),
  kappa_by_role = c(adult = 1, child = 1.1, elderly = 0.7, toddler = 1.2),
  latent_shape = 2,
  latent_scale = 1,
  infectious_shape = 2,
  infectious_scale = 2,
```

```
  peak_day = 2,
  width = 2,
  max_days = 40,
  test_weekly_before_detection = TRUE,
  perfect_detection = TRUE,
  contact_mat = NULL,
  verbose = FALSE,
  seasonal_forcing_list = NULL
)
```

Arguments

hh_id	Household identifier.
roles	Character vector of roles (one per person).
alpha_comm_by_role	Numeric; baseline community hazard (scalar).
beta1, beta2, delta, rho, V_ref	Numeric; transmission constants.
phi_by_role, kappa_by_role	Named numeric vectors; susceptibility and infectivity multipliers by role.
latent_shape, latent_scale, infectious_shape, infectious_scale	Numeric; gamma parameters for latent/infectious durations.
peak_day, width	Numeric; infectivity profile controls.
max_days	Integer; simulation horizon (days).
test_weekly_before_detection	Logical; weekly testing before first detection.
perfect_detection	Logical; perfect detection on test days.
contact_mat	Optional square contact matrix (no self-contact).
verbose	Logical; print progress.
seasonal_forcing_list	Optional list of length 4 with role-specific seasonal multipliers (length max_days).

Value

A data frame with columns hh_id, person_id, role, infection_time, infectious_start, infectious_end, detection_time, infection_resolved, plus list-columns vl_full_trajectory, viral_loads_test_days. Attributes test_days and params are attached.

---

simulate_viral_load_trajectory

Simulate a log10 viral-load trajectory

Description

Generates a unimodal log10 viral-load curve given peak level/day and growth/decay rates.

Usage

```
simulate_viral_load_trajectory(t, v_p, t_p, lambda_g, lambda_d)
```

Arguments

t	Integer vector; days relative to infection.
v_p	Numeric; peak log10 VL.
t_p	Integer; day of peak.
lambda_g, lambda_d	Numeric; growth/decay rates.

Value

Numeric vector of log10 VL at times t.

---

summarize_individuals *Summarize individual-level infection data*

Description

Produces one row per individual with detection windows, inferred infectious windows (relative days), index-case flags, observation bounds, role/age classification, and aggregated covariates. Optionally builds day-series list-columns for selected covariates.

Usage

```
summarize_individuals(
  raw_dt,
  study_start,
  study_end,
  day_series_covariates = TRUE,
  series_cols = NULL
)
```

Arguments

raw_dt	List of household-level data frames/tables.
study_start, study_end	Date. Analysis window defining day indices.
day_series_covariates	Logical; add day-series list-columns (default TRUE).
series_cols	Character or NULL; covariates to series-encode (default NULL = all).

Value

data.table with one row per individual containing detection and infectious windows, flags, observation bounds, role/age, and covariate summaries.

`TransmissionChainAnalysis` *TransmissionChainAnalysis: analyze user-supplied household data*

Description

Runs the end-to-end workflow on *user-provided* data using either the legacy MLE path (`estimation_method = "mle"`) or the RSV/VL Stan path (`estimation_method = "stan"`). This function does not simulate data; it expects input in tabular form (see Details).

Usage

```
TransmissionChainAnalysis(
  user_data,
  estimation_method = c("mle", "stan"),
  data_summary = FALSE,
  postprocessing = TRUE,
  plots = c("daily", "weekly", "timeline", "sar"),
  print_plots = FALSE,
  index_vl_column = "vl_test",
  n_runs = 10,
  day_series_covariates = TRUE,
  series_cols = NULL,
  comm_covariate_cols = NULL,
  hh_covariate_cols = NULL,
  hh_by_role = FALSE,
  hh_role_covariate_cols = NULL,
  standardize_covariates = TRUE,
  lambda_comm = 0.01,
  lambda_hh = 0.01,
  start_date = as.Date("2024-09-21"),
  end_date = as.Date("2025-04-17"),
  latent_par = list(shape = 2, scale = 1),
  report_par = list(shape = 1, scale = 1.5),
  infect_par = list(shape = 3, scale = 2),
  start_par = c(-6, 0.02, -2, rep(0, 6)),
  lambda = 0.01,
  lambda0 = 0.2,
  lambda_alpha = 5,
  delta0_true = qlogis(0.002),
  alpha0_true = qlogis(0.2),
  true_values = NULL,
  seasonal_forcing_list = NULL,
  max_days = 365,
  stan_file = "../inst/stan/HH_parameter_estimation2.stan",
  stan_chains = 1,
  stan_iter = 100,
  stan_warmup = 100,
  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.
`estimation_method`	One of `c("mle","stan")`. Valid pairs are legacy+MLE (`"mle"`) and RSV/VL+Stan (`"stan"`).
`data_summary`	Logical; print/retain a brief summary in print methods (see Value).
`postprocessing`	Logical; for MLE, compute a post-processing table; for Stan, this is an alias of the posterior summary.
`plots`	Character vector of plot names to build when compatible data are present: `c("daily","weekly","ti` or `"all"`.
`print_plots`	Logical; print plots if produced.
`index_vl_column`	Optional character; name of the viral-load column used by SAR-by-VL plotting. Defaults to `"vl_test"` when present.
`n_runs`	Integer; number of repeated estimation runs (MLE path only).
`day_series_covariates`	Logical; include day-series covariates (MLE path).
`series_cols`	Optional character vector; selected day-series columns (MLE path).
`comm_covariate_cols`, `hh_covariate_cols`	Optional character vectors of likelihood-level covariates (MLE path).
`hh_by_role`	Logical; fit separate household effects by role (MLE path).
`hh_role_covariate_cols`	Optional named list of role-specific covariates (MLE path).
`standardize_covariates`	Logical; standardize covariates (MLE path).
`lambda_comm`, `lambda_hh`	Numeric; ridge penalties for community and household covariates (MLE path).
`start_date`, `end_date`	`Date` study window.
`latent_par`, `report_par`, `infect_par`	Lists of gamma parameters (shape, scale) used by imputation (MLE path).
`start_par`	Numeric vector; optimizer starting values (MLE path).
`lambda`, `lambda0`, `lambda_alpha`	Numeric; optimizer penalties (MLE path).
`delta0_true`, `alpha0_true`	Numeric anchors on the logit scale (MLE path).
`true_values`	Optional named numeric vector of reference/true values for post-processing (MLE path).

`seasonal_forcing_list`
: Optional named list of role vectors for forcing (Stan path).

`max_days`
: Integer; maximum modeled days (Stan path).

`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 for Stan path.

`vl_source`
: One of `c("column","simulate","none")` indicating the VL source for Stan path.

`vl_column`
: Optional character; the VL column when `vl_source="column"` (Stan path).

`role_levels`
: Character vector; canonical role levels for normalization on the Stan path.

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.

Only two engine/method pairs are supported: legacy+MLE and RSV/VL+Stan.

Value

An object of class `"TransmissionChainResult"` containing:

- `$results`: engine-specific internals (MLE: summaries/person-day/estimates; Stan: stan_data, fit, posterior_summary).
- `$postprocessing`: MLE summary table or Stan posterior summary.
- `$plot_list`: named list of ggplot objects when built.
- `$data_summary`: flag indicating whether print methods preview summaries.

See Also

`main_parameter_estimation_pipeline`, `postprocessing_estimates`

Examples

```
## Not run:
# MLE path requires full legacy preprocessing; shown here schematically:
# res <- TransmissionChainAnalysis(user_data = your_long_df, estimation_method = "mle")

## End(Not run)
```

Index

attach_vl_full_trajectory_from_long, 2

build_person_day_table, 3, *22*
build_stan_household_arrays, 4, *17*, *23*

data_summarization, 5
dataframe_to_household_list, 5

g_rescaled, 12
generate_household_roles, 6
generate_synthetic_data_one, 7
GenSyn, 9, *20*

households_to_long_tests, 13

infectious_time_imputation, *3*, 14

main_parameter_estimation_pipeline, *6*, *9*, *10*, *12*, 14, *33*

normalize_roles, 18

p_transmission, 21
postprocess_stan_fit, 19
postprocessing_estimates, *12*, *17*, 18, *33*
prepare_stan_households_from_user_data, 19
print.GenSynResult, 20
print.TransmissionChainResult, 21

run_household_stan, *17*, *19*, 23
running_parameter_estimation, 22

seasonal_cosine, 24
seasonal_forcing_to_series, 24
simulate_households, 25
simulate_multiple_households_comm, 27
simulate_one_household_comm, *27*, 28
simulate_viral_load_trajectory, 29
summarize_individuals, *6*, *14*, 30

TransmissionChainAnalysis, *16*, *17*, *21*, 31