Package 'mitey'

September 2, 2025

Title Serial Interval and Case Reproduction Number Estimation

Version 0.2.0

Description Provides methods to estimate serial intervals and time-varying case reproduction numbers from infectious disease outbreak data. Serial intervals measure the time between symptom onset in linked transmission pairs, while case reproduction numbers quantify how many secondary cases each infected individual generates over time. These parameters are essential for understanding transmission dynamics, evaluating control measures, and informing public health responses. The package implements the maximum likelihood framework from Vink et al. (2014) <doi:10.1093/aje/kwu209> for serial interval estimation and the retrospective method from Wallinga & Lipsitch (2007) <doi:10.1098/rspb.2006.3754> for reproduction number estimation. Originally developed for scabies transmission analysis but applicable to other infectious diseases including influenza, COVID-19, and emerging pathogens. Designed for epidemiologists, public health researchers, and infectious disease modelers working with outbreak surveillance data.

Date 2025-08-16

License EUPL-1.2

Encoding UTF-8

RoxygenNote 7.3.2

Imports fdrtool, ggplot2, stats

URL https://github.com/kylieainslie/mitey, https://kylieainslie.github.io/mitey/

BugReports https://github.com/kylieainslie/mitey/issues

Suggests brms, broom, cowplot, dplyr, EpiLPS, flextable, forcats, ftExtra, ggridges, gt, here, ISOweek, knitr, lubridate, officer, openxlsx, outbreaks, purrr, rmarkdown, stringr, testthat (>= 3.0.0), tidybayes, tidyr, viridis, zoo

Config/testthat/edition 3

VignetteBuilder knitr

Depends R (>= 3.5.0)

NeedsCompilation no

Author Kylie Ainslie [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-5419-7206>)

Maintainer Kylie Ainslie <ainslie.kylie@gmail.com>

Repository CRAN

Date/Publication 2025-09-02 05:40:18 UTC

Contents

`calculate_bootstrap_ci` *Calculate Bootstrap Confidence Intervals for R Estimates*

Description

Generates bootstrap confidence intervals for reproduction number estimates by resampling the incidence data multiple times and calculating quantiles of the resulting R distributions.

Usage

```
calculate_bootstrap_ci(
  incidence,
  si_prob,
  dates,
  si_mean,
  si_sd,
  si_dist,
  smoothing,
  n_bootstrap,
  conf_level
)
```

Arguments

`incidence`	numeric vector; daily case counts
`si_prob`	numeric matrix; serial interval probability matrix
`dates`	vector; dates corresponding to incidence data
`si_mean`	numeric; mean of the serial interval distribution
`si_sd`	numeric; standard deviation of the serial interval distribution
`si_dist`	character; distribution type, either "gamma" or "normal"
`smoothing`	integer; window size for temporal smoothing
`n_bootstrap`	integer; number of bootstrap samples to generate
`conf_level`	numeric; confidence level (between 0 and 1)

Value

named list with confidence interval bounds:

- `r_lower`, `r_upper`: Confidence intervals for raw R estimates
- `r_corrected_lower`, `r_corrected_upper`: Confidence intervals for corrected R estimates

`calculate_r_estimates` *Calculate Reproduction Number Estimates*

Description

Implements the Wallinga-Lipsitch algorithm to estimate case reproduction numbers from incidence data and serial interval probabilities. This function performs the likelihood calculations for retrospective reproduction number estimation.

Usage

```
calculate_r_estimates(
  incidence,
  si_prob,
  dates,
  si_mean,
  si_sd,
  si_dist,
  smoothing
)
```

Arguments

`incidence`	numeric vector; daily case counts. Must be non-negative integers. Days with zero cases will have R estimates of NA
`si_prob`	numeric matrix; serial interval probability matrix from `calculate_si_probability_matrix`. Element `[i, j]` represents the probability that case j infected case i
`dates`	vector; dates corresponding to incidence data. Used for right-truncation correction calculations
`si_mean`	numeric; mean of the serial interval distribution in days
`si_sd`	numeric; standard deviation of the serial interval distribution in days
`si_dist`	character; distribution type for serial interval, either "gamma" or "normal"
`smoothing`	integer; window size for temporal smoothing (0 = no smoothing). When > 1, applies centered moving average to reduce noise

Details

The algorithm calculates the probability that each earlier case infected each later case based on their time difference and the serial interval distribution. These probabilities are then aggregated to estimate the expected number of secondary cases generated by cases on each day.

The Wallinga-Lipsitch method works by:

1. Computing transmission likelihoods from earlier to later cases
2. Normalizing these likelihoods to create proper probabilities
3. Aggregating probabilities to estimate expected secondary cases per primary case
4. Applying right-truncation correction for cases near the observation end

The right-truncation correction accounts for the fact that cases near the end of the observation period may have generated secondary cases that occur after data collection ended.

Value

named list with two numeric vectors of the same length as `incidence`:

- `r`: Raw case reproduction number estimates. Returns NA for days with zero cases or single-case epidemics
- `r_corrected`: Estimates with right-truncation correction applied. Values > 10 are capped at NA to avoid unrealistic estimates

See Also

wallinga_lipsitch for the main user interface, calculate_si_probability_matrix for probability matrix creation, calculate_truncation_correction for correction details

---

`calculate_si_probability_matrix`
Calculate Serial Interval Probability Matrix

Description

Computes a matrix of transmission probabilities between all pairs of cases based on their time differences and the specified serial interval distribution. Only considers epidemiologically plausible transmission pairs (earlier to later cases).

Usage

```
calculate_si_probability_matrix(day_diffs, si_mean, si_sd, si_dist)
```

Arguments

`day_diffs`	numeric matrix; matrix of day differences between each pair of cases, where element `[i,j]` represents days between case i and case j
`si_mean`	numeric; mean of the serial interval distribution in days
`si_sd`	numeric; standard deviation of the serial interval distribution in days
`si_dist`	character; distribution type, either "gamma" or "normal"

Value

numeric matrix; matrix of transmission probabilities where element `[i,j]` represents the probability that case j infected case i based on their time difference and the serial interval distribution

Examples

```
# Create sample day differences matrix
dates <- as.Date(c("2023-01-01", "2023-01-03", "2023-01-05"))
day_diffs <- create_day_diff_matrix(dates)

# Calculate probability matrix
prob_matrix <- calculate_si_probability_matrix(day_diffs, si_mean = 7, si_sd = 3, si_dist = "gamma")
```

`calculate_truncation_correction` *Calculate Right-Truncation Correction Factors*

Description

Computes correction factors to adjust reproduction number estimates for right-truncation bias. This bias occurs because cases near the end of the observation period may have generated secondary cases that are not yet observed.

Usage

```
calculate_truncation_correction(dates, si_mean, si_sd, si_dist)
```

Arguments

`dates`	vector; dates corresponding to each case
`si_mean`	numeric; mean of the serial interval distribution in days
`si_sd`	numeric; standard deviation of the serial interval distribution in days
`si_dist`	character; distribution type, either "gamma" or "normal"

Value

numeric vector; correction factors for each case. Values > 1 indicate upward adjustment needed. Returns NA when correction would be unreliable (probability of observation <= 0.5)

Examples

```
# Calculate truncation correction for recent cases
case_dates <- seq(as.Date("2023-01-01"), as.Date("2023-01-20"), by = "day")
corrections <- calculate_truncation_correction(
  case_dates, si_mean = 7, si_sd = 3, si_dist = "gamma"
  )

# Show how correction increases for more recent cases
tail(corrections, 5)
```

`conv_tri_dist` *Convolution of the triangular distribution with the mixture component density (continuous case)*

Description

We split the folded normal distribution for Primary-Secondary, Primary-Tertiary and Primary-Quaternary routes into two parts

- component 1: Co-Primary route
- component 2+3: Primary-Secondary route
- component 4+5: Primary-Tertiary route
- component 6+7: Primary-Quaternary route

Usage

```
conv_tri_dist(x, sigma = sd(x), r = x, mu = mean(x), route, quantity = "zero")
```

Arguments

`x`	vector of index case to case intervals
`sigma`	standard deviation of density distribution
`r`	description??
`mu`	mean of density distribution
`route`	integer; between 1 and 7 and indicates the route of transmission.
`quantity`	character; "zero", "lower", "upper"

Value

vector of density draws for each value of x

Examples

```
iccs <- 1:30
conv_tri_dist(x = iccs, route = 1)
```

create_day_diff_matrix *Create Day Difference Matrix*

Description

Creates a symmetric matrix containing the time differences (in days) between all pairs of cases based on their symptom onset dates.

Usage

```
create_day_diff_matrix(dates)
```

Arguments

dates	vector; dates of symptom onset for each case. Can be Date objects or any format coercible to dates

Value

numeric matrix; symmetric matrix where element `[i,j]` represents the number of days between case i and case j (positive if i occurs after j)

Examples

```
# Create day difference matrix from onset dates
onset_dates <- as.Date(c("2023-01-01", "2023-01-04", "2023-01-07", "2023-01-10"))
day_differences <- create_day_diff_matrix(onset_dates)
print(day_differences)
```

f0 *Calculate f0 for Different Components*

Description

This function calculates the value of f0 based on the component, where the components represent the transmission routes: Co-Primary (CP), Primary-Secondary (PS), Primary-Tertiary (PT), and Primary-Quaternary (PQ). We split the PS, PT and PQ routes into two parts, such that

- component 1: CP route
- component 2+3: PS route
- component 4+5: PT route
- component 6+7: PQ route

Usage

```
f0(x, mu, sigma, comp, dist = "normal")
```

Arguments

x	numeric; the value at which to evaluate the function.
mu	numeric; the mean value.
sigma	numeric; the standard deviation.
comp	integer; the component number (1 to 7).
dist	string; assumed distribution of the serial interval; takes "normal" or "gamma"; defaults to "normal"

Details

If the dist = gamma, then the mean (μ) and standard deviation (sigma) are converted into the shape (k) and scale (theta) parameters for the gamma distribution, such that the mean (μ) and variance (σ^2) are given by:

$$\mu = k \times \theta$$

$$\sigma^2 = k \times \theta^2$$

.

Value

The calculated value of f0.

Examples

```
# Basic example with normal distribution
# Component 2 represents primary-secondary transmission
f0(x = 0.5, mu = 12, sigma = 3, comp = 2, dist = "normal")

# Same parameters with gamma distribution
f0(x = 0.5, mu = 12, sigma = 3, comp = 2, dist = "gamma")

# Component 1 represents co-primary transmission
f0(x = 0.3, mu = 8, sigma = 2, comp = 1, dist = "normal")

# Calculate for all transmission route components
x_val <- 0.4
mu_val <- 10
sigma_val <- 3

# Components 1-7 represent different transmission routes:
# 1: Co-Primary, 2+3: Primary-Secondary, 4+5: Primary-Tertiary, 6+7: Primary-Quaternary
sapply(1:7, function(comp) {
  f0(x_val, mu_val, sigma_val, comp, "normal")
})
```

flower *Calculate flower for Different Components*

Description

This function calculates the value of flower based on the component.

Usage

```
flower(x, r, mu, sigma, comp, dist = "normal")
```

Arguments

x	The value at which to evaluate the function.
r	The value of r.
mu	The mean value.
sigma	The standard deviation.
comp	The component number (1 to 7).
dist	string; assumed distribution of the serial interval; accepts "normal" or "gamma"; defaults to "normal"

Value

The calculated value of flower.

Examples

```
# Basic example with normal distribution
# Component 2 represents primary-secondary transmission
flower(x = 15, r = 10, mu = 12, sigma = 3, comp = 2, dist = "normal")

# Same parameters with gamma distribution
flower(x = 15, r = 10, mu = 12, sigma = 3, comp = 2, dist = "gamma")

# Component 1 represents co-primary transmission
flower(x = 5, r = 20, mu = 8, sigma = 2, comp = 1, dist = "normal")

# Calculate for all transmission route components
x_val <- 20
r_val <- 25
mu_val <- 10
sigma_val <- 3

# Components 1-7 represent different transmission routes
sapply(1:7, function(comp) {
  flower(x_val, r_val, mu_val, sigma_val, comp, "normal")
})
```

fupper *Calculate fupper for Different Components*

Description

This function calculates the value of fupper based on the component.

Usage

```
fupper(x, r, mu, sigma, comp, dist = "normal")
```

Arguments

x	The value at which to evaluate the function.
r	The value of r.
mu	The mean value.
sigma	The standard deviation.
comp	The component number (1 to 7).
dist	string; assumed distribution of the serial interval; accepts "normal" or "gamma"; defaults to "normal".

Value

The calculated value of fupper.

Examples

```
# Basic example with normal distribution
# Component 2 represents primary-secondary transmission
fupper(x = 15, r = 20, mu = 12, sigma = 3, comp = 2, dist = "normal")

# Same parameters with gamma distribution
fupper(x = 15, r = 20, mu = 12, sigma = 3, comp = 2, dist = "gamma")

# Component 1 represents co-primary transmission
fupper(x = 5, r = 25, mu = 8, sigma = 2, comp = 1, dist = "normal")

# Calculate for all transmission route components
x_val <- 10
r_val <- 30
mu_val <- 12
sigma_val <- 3

# Components 1-7 represent different transmission routes
sapply(1:7, function(comp) {
  fupper(x_val, r_val, mu_val, sigma_val, comp, "normal")
})
```

`f_gam` *Calculate serial interval mixture density assuming underlying gamma distribution*

Description

This function computes the weighted mixture density for serial intervals based on different transmission routes in an outbreak. It implements part of the Vink et al. (2014) method for serial interval estimation, assuming an underlying gamma distribution for the serial interval.

Usage

```
f_gam(x, w1, w2, w3, mu, sigma)
```

Arguments

`x`	quantile or vector of quantiles (time in days since index case symptom onset)
`w1`	probability weight of being a co-primary case
`w2`	probability weight of being a primary-secondary case
`w3`	probability weight of being a primary-tertiary case
`mu`	mean serial interval in days (must be positive)
`sigma`	standard deviation of serial interval in days (must be positive)

Details

The function models four distinct transmission routes:

- Co-primary (CP): Cases infected simultaneously from the same source
- Primary-secondary (PS): Direct transmission from index case
- Primary-tertiary (PT): Transmission through one intermediate case
- Primary-quaternary (PQ): Transmission through two intermediate cases

Each route contributes to the overall serial interval distribution with different means and variances. The co-primary component uses a modified gamma distribution to account for simultaneous infections, while subsequent generations follow gamma distributions with progressively longer means and larger variances.

This function is primarily used internally by `si_estim` when `dist = "gamma"` is specified, and by `plot_si_fit` for visualizing fitted distributions.

The weights w1, w2, and w3 must sum to <= 1, with the remaining probability (1 - w1 - w2 - w3) assigned to primary-quaternary cases. The function converts the mean and standard deviation to gamma distribution shape (k) and scale (\theta) parameters using the method of moments:

$$k = \mu^2/\sigma^2$$

$$\theta = \sigma^2/\mu$$

Value

Vector of weighted density values corresponding to input quantiles x. Returns the sum of densities from all four transmission routes.

References

Vink, M. A., Bootsma, M. C. J., & Wallinga, J. (2014). Serial intervals of respiratory infectious diseases: A systematic review and analysis. American Journal of Epidemiology, 180(9), 865-875.

See Also

si_estim, plot_si_fit, f_norm

Examples

```
# Example: Plot serial interval mixture density for influenza-like outbreak

# Set parameters for a typical respiratory infection
mu <- 6.5      # Mean serial interval of 6.5 days
sigma <- 2.8   # Standard deviation of 2.8 days

# Set transmission route weights
w1 <- 0.1      # 10% co-primary cases
w2 <- 0.6      # 60% primary-secondary cases
w3 <- 0.2      # 20% primary-tertiary cases
# Remaining 10% are primary-quaternary cases (1 - w1 - w2 - w3 = 0.1)

# Create sequence of time points
x <- seq(0.1, 30, by = 0.1)

# Calculate mixture density
density_values <- f_gam(x, w1, w2, w3, mu, sigma)

# Plot the result
plot(x, density_values, type = "l", lwd = 2, col = "red",
     xlab = "Days", ylab = "Density",
     main = "Serial Interval Mixture Density (Gamma Distribution)")
grid()
```

---

f_norm *Calculate serial interval mixture density assuming underlying normal distribution*

Description

This function computes the weighted mixture density for serial intervals based on different transmission routes in an outbreak. It implements part of the Vink et al. (2014) method for serial interval estimation, assuming an underlying normal distribution for the serial interval.

Usage

```
f_norm(x, w1, w2, w3, mu, sigma)
```

Arguments

x	quantile or vector of quantiles (time in days since index case symptom onset)
w1	probability weight of being a co-primary case
w2	probability weight of being a primary-secondary case
w3	probability weight of being a primary-tertiary case
mu	mean serial interval in days (can be any real number)
sigma	standard deviation of serial interval in days (must be positive)

Details

The function models four distinct transmission routes:

- Co-primary (CP): Cases infected simultaneously from the same source
- Primary-secondary (PS): Direct transmission from index case
- Primary-tertiary (PT): Transmission through one intermediate case
- Primary-quaternary (PQ): Transmission through two intermediate cases

Each route contributes to the overall serial interval distribution with different means and variances. The co-primary component uses a half-normal distribution to model simultaneous infections (preventing negative serial intervals), while subsequent generations follow normal distributions with means that are multiples of the base serial interval.

This function is primarily used internally by `si_estim` when `dist = "normal"` is specified (the default), and by `plot_si_fit` for visualizing fitted distributions. The normal distribution assumption allows for negative serial intervals, which may be more realistic for some pathogens.

The weights w1, w2, and w3 must sum to <= 1, with the remaining probability (1 - w1 - w2 - w3) assigned to primary-quaternary cases. The transmission route distributions are parameterized as: Co-primary: Half-normal with scale parameter derived from sigma Primary-secondary: Normal(mu, sigma) Primary-tertiary: Normal(2*mu, sqrt(2)*sigma) Primary-quaternary: Normal(3*mu, sqrt(3)*sigma)

Value

Vector of weighted density values corresponding to input quantiles x. Returns the sum of densities from all four transmission routes.

References

Vink, M. A., Bootsma, M. C. J., & Wallinga, J. (2014). Serial intervals of respiratory infectious diseases: A systematic review and analysis. American Journal of Epidemiology, 180(9), 865-875.

See Also

`si_estim`, `plot_si_fit`, `f_gam`

Examples

```
# Example: Plot serial interval mixture density for scabies outbreak

# Set parameters based on scabies epidemiology (longer serial interval)
mu <- 123     # Mean serial interval of 123 days (from Ainslie et al.)
sigma <- 32   # Standard deviation of 32 days

# Set transmission route weights typical for scabies
w1 <- 0.15    # 15% co-primary cases
w2 <- 0.50    # 50% primary-secondary cases
w3 <- 0.25    # 25% primary-tertiary cases
# Remaining 10% are primary-quaternary cases (1 - w1 - w2 - w3 = 0.1)

# Create sequence of time points
x <- seq(0, 400, by = 1)

# Calculate mixture density
density_values <- f_norm(x, w1, w2, w3, mu, sigma)

# Plot the result
plot(x, density_values, type = "l", lwd = 2, col = "red",
     xlab = "Days", ylab = "Density",
     main = "Serial Interval Mixture Density (Normal Distribution)")
grid()
```

---

generate_case_bootstrap

Generate Bootstrap Sample of Case Incidence

Description

Creates a bootstrap sample by resampling individual cases with replacement, then reconstructing daily incidence counts. This maintains the temporal distribution while introducing sampling variation for uncertainty estimation.

Usage

```
generate_case_bootstrap(incidence)
```

Arguments

incidence	numeric vector; daily case counts (non-negative integers)

Value

numeric vector; bootstrapped daily incidence of the same length as input. Total number of cases remains the same but their temporal distribution varies

`generate_synthetic_epidemic` *Generate Synthetic Epidemic Data Using the Renewal Equation*

Description

Simulates epidemic incidence data with known reproduction numbers using the renewal equation framework. This function is useful for testing and validating reproduction number estimation methods, as it generates synthetic outbreaks with ground truth R values that can be compared against estimated values.

Usage

```
generate_synthetic_epidemic(
  true_r,
  si_mean,
  si_sd,
  si_dist = "gamma",
  initial_cases = 10
)
```

Arguments

`true_r`	numeric vector; the true time-varying reproduction numbers. The length of this vector determines the number of days in the simulated epidemic
`si_mean`	numeric; the mean of the serial interval distribution in days
`si_sd`	numeric; the standard deviation of the serial interval distribution in days
`si_dist`	character; the distribution family for the serial interval. Must be either "gamma" (default) or "normal". Gamma is recommended as it naturally restricts to positive values
`initial_cases`	integer; the number of cases on the first day of the epidemic. Defaults to 10

Details

The function implements the discrete renewal equation:

$$\lambda_t = \sum_{s=1}^{t-1} I_s \cdot R_s \cdot w(t-s)$$

where λ_t is the expected number of new infections at time t, I_s is the incidence at time s, R_s is the reproduction number at time s, and $w(t-s)$ is the probability mass function of the serial interval distribution for interval $t-s$.

New cases at each time point are drawn from a Poisson distribution with mean λ_t, introducing realistic stochastic variation while maintaining the specified reproduction number trajectory.

The serial interval distribution is truncated at the 99th percentile to avoid computationally expensive calculations for very long tails. For the normal distribution, the probability mass function is normalized to ensure proper probability weights.

This function is particularly useful for:

- Validating reproduction number estimation methods
- Testing the performance of epidemiological models
- Generating realistic epidemic scenarios for research
- Creating training data for machine learning approaches

Value

A data frame with three columns:

- `date`: Date sequence starting from "2023-01-01"
- `true_r`: The input reproduction number values
- `incidence`: Simulated daily case counts

References

Fraser C (2007). Estimating individual and household reproduction numbers in an emerging epidemic. PLoS One, 2(8), e758.

Cori A, Ferguson NM, Fraser C, Cauchemez S (2013). A new framework and software to estimate time-varying reproduction numbers during epidemics. American Journal of Epidemiology, 178(9), 1505-1512.

See Also

`wallinga_lipsitch` for reproduction number estimation methods that can be applied to the generated data

Examples

```
# Simple epidemic with constant R = 1.5
constant_r <- rep(1.5, 30)
epidemic1 <- generate_synthetic_epidemic(
  true_r = constant_r,
  si_mean = 7,
  si_sd = 3,
  si_dist = "gamma"
)
head(epidemic1)

# Epidemic with declining R (e.g., intervention effect)
declining_r <- seq(2.0, 0.5, length.out = 50)
epidemic2 <- generate_synthetic_epidemic(
  true_r = declining_r,
  si_mean = 5,
  si_sd = 2,
```

```
  si_dist = "gamma",
  initial_cases = 5
)

# Epidemic with seasonal pattern
days <- 100
seasonal_r <- 1.2 + 0.5 * sin(2 * pi * (1:days) / 365 * 7) # Weekly seasonality
epidemic3 <- generate_synthetic_epidemic(
  true_r = seasonal_r,
  si_mean = 6,
  si_sd = 2.5,
  si_dist = "normal"
)

# Plot the results
if (require(ggplot2)) {
  library(ggplot2)
  ggplot(epidemic1, aes(x = date)) +
    geom_col(aes(y = incidence), alpha = 0.7) +
    geom_line(aes(y = true_r * 10), color = "red") +
    labs(title = "Synthetic Epidemic",
         y = "Daily Incidence",
         subtitle = "Red line: True R × 10")
}
```

`integrate_component` *Integrate Serial Interval Component Functions for Likelihood Calculation*

Description

This function performs numerical integration of serial interval component functions used in the Vink method for estimating serial interval distributions. It integrates the probability density functions for different transmission routes over specified intervals as part of the Expectation-Maximization algorithm.

Usage

```
integrate_component(
  d,
  mu,
  sigma,
  comp,
  dist = c("normal", "gamma"),
  lower = TRUE
)
```

Arguments

d	numeric; the index case-to-case (ICC) interval in days for which to calculate the likelihood contribution
mu	numeric; the mean of the serial interval distribution in days
sigma	numeric; the standard deviation of the serial interval distribution in days
comp	integer; the transmission route component number (1 to 7). See Details for component definitions
dist	character; the assumed underlying distribution of the serial interval. Must be either "normal" or "gamma". Defaults to "normal"
lower	logical; if TRUE (default), performs integration using flower and fupper functions. If FALSE, uses f0 function

Details

The function supports two integration modes:

- lower = TRUE: Integrates using flower and fupper functions over intervals [d-1, d] and [d, d+1] respectively, representing the likelihood contribution when case occurs at day d
- lower = FALSE: Integrates using f0 function over interval [d, d+1], representing an alternative likelihood formulation

The components represent different transmission routes in outbreak analysis:

- Component 1: Co-Primary (CP) transmission
- Components 2+3: Primary-Secondary (PS) transmission
- Components 4+5: Primary-Tertiary (PT) transmission
- Components 6+7: Primary-Quaternary (PQ) transmission

This function is primarily used internally by si_estim() as part of the Vink method for estimating serial interval parameters from outbreak data.

Value

numeric; the integrated likelihood value for the specified component and data point. Used in the EM algorithm for serial interval estimation

References

Vink MA, Bootsma MCJ, Wallinga J (2014). Serial intervals of respiratory infectious diseases: A systematic review and analysis. American Journal of Epidemiology, 180(9), 865-875.

See Also

flower, fupper, f0, si_estim

Examples

```
# Basic example with lower integration (default)
# Component 2 represents primary-secondary transmission
integrate_component(d = 15, mu = 12, sigma = 3, comp = 2, dist = "normal", lower = TRUE)

# Upper integration example
integrate_component(d = 15, mu = 12, sigma = 3, comp = 2, dist = "normal", lower = FALSE)

# Using gamma distribution
integrate_component(d = 10, mu = 8, sigma = 2, comp = 1, dist = "gamma", lower = TRUE)

# Component 1 (co-primary transmission) with normal distribution
integrate_component(d = 5, mu = 10, sigma = 3, comp = 1, dist = "normal", lower = TRUE)

# Compare different components for the same data point
d_val <- 20
mu_val <- 15
sigma_val <- 4

# Calculate for components 1, 2, and 4 (different transmission routes)
sapply(c(1, 2, 4), function(comp) {
  integrate_component(d_val, mu_val, sigma_val, comp, "normal", lower = TRUE)
})
```

---

integrate_components_wrapper

Compute Serial Interval Component Integrals for All Transmission Routes

---

Description

This wrapper function efficiently computes the likelihood contributions for all relevant transmission route components for a given index case-to-case (ICC) interval. It is a key component of the Vink method's Expectation-Maximization algorithm for estimating serial interval parameters from outbreak data.

Usage

```
integrate_components_wrapper(d, mu, sigma, dist = "normal")
```

Arguments

d	numeric; the index case-to-case (ICC) interval in days. Represents the time difference between the symptom onset of the index case (latest case) and the current case being evaluated. Must be non-negative
mu	numeric; the mean of the serial interval distribution in days. Must be positive for meaningful epidemiological interpretation

`sigma` numeric; the standard deviation of the serial interval distribution in days. Must be positive

`dist` character; the assumed underlying distribution family for the serial interval. Must be either "normal" or "gamma". Defaults to "normal". Gamma distribution is often preferred for serial intervals as it naturally restricts to positive values

Details

The function handles different integration scenarios based on the distribution type and ICC interval value:

- For **normal distribution**: Uses all 7 components representing the full mixture of transmission routes (co-primary, primary-secondary with positive and negative components, primary-tertiary, and primary-quaternary routes)
- For **gamma distribution**: Uses components 1, 2, 4, and 6 only, as the gamma distribution naturally handles only positive serial intervals, eliminating the need for negative component pairs
- For **ICC interval = 0**: Uses upper integration (`lower = FALSE`) representing the special case of simultaneous symptom onset
- For **ICC interval > 0**: Uses lower integration (`lower = TRUE`) representing the standard transmission likelihood calculation

This function is primarily used internally by `si_estim()` as part of the E-step in the EM algorithm. Each component represents a different hypothesis about the transmission route:

- Component 1: Co-primary transmission (simultaneous exposure)
- Components 2-3: Primary-secondary transmission (direct transmission)
- Components 4-5: Primary-tertiary transmission (second generation)
- Components 6-7: Primary-quaternary transmission (third generation)

For gamma distributions, components 3, 5, and 7 are omitted because the gamma distribution naturally handles the asymmetry that these components would otherwise model in the normal distribution case.

Value

numeric vector; integrated likelihood values for each relevant transmission route component. The length depends on the distribution:

- Normal distribution: 7 values (components 1-7)
- Gamma distribution: 4 values (components 1, 2, 4, 6)

References

Vink MA, Bootsma MCJ, Wallinga J (2014). Serial intervals of respiratory infectious diseases: A systematic review and analysis. American Journal of Epidemiology, 180(9), 865-875.

See Also

integrate_component, si_estim, flower, fupper, f0

Examples

```
# Basic example with normal distribution
# Returns 7 component values for ICC interval of 10 days
integrate_components_wrapper(d = 10, mu = 15, sigma = 3, dist = "normal")

# Same parameters with gamma distribution
# Returns 4 component values (components 1, 2, 4, 6)
integrate_components_wrapper(d = 10, mu = 15, sigma = 3, dist = "gamma")

# Special case: ICC interval of 0 (simultaneous onset)
integrate_components_wrapper(d = 0, mu = 12, sigma = 2, dist = "normal")
```

---

plot_si_fit *Visualize Serial Interval Distribution Fit to Outbreak Data*

Description

Creates a diagnostic plot showing the fitted serial interval mixture distribution overlaid on a histogram of observed index case-to-case (ICC) intervals from outbreak data.

Usage

```
plot_si_fit(dat, mean, sd, weights, dist = "normal", scaling_factor = 1)
```

Arguments

dat	numeric vector; the index case-to-case (ICC) intervals in days. These represent the time differences between symptom onset in the index case (case with earliest symptom onset) and each other case in the outbreak
mean	numeric; the estimated mean of the serial interval distribution in days, typically obtained from si_estim()
sd	numeric; the estimated standard deviation of the serial interval distribution in days, typically obtained from si_estim()
weights	numeric vector; the estimated weights for different transmission route components. Length and interpretation depends on distribution: • **Normal distribution**: 4 weights corresponding to aggregated transmission routes (co-primary, primary-secondary, primary-tertiary, primary-quaternary) • **Gamma distribution**: 3 weights for the reduced component set
dist	character; the distribution family used for serial interval estimation. Must be either "normal" (default) or "gamma". Should match the distribution used in the original si_estim() call

`scaling_factor` numeric; multiplicative factor to adjust the height of the fitted density curve relative to the histogram. Values > 1 make the curve higher, values < 1 make it lower. Defaults to 1. Useful when histogram and density have different scales.

Details

The function displays:

- **Histogram**: Observed ICC intervals binned by day, representing the empirical distribution of time differences between symptom onset in the index case and all other cases in the outbreak
- **Fitted curve**: The estimated mixture distribution combining different transmission routes (co-primary, primary-secondary, primary-tertiary, and primary-quaternary), weighted according to their estimated probabilities
- **Reference line**: For normal distributions, a dashed vertical line indicates the estimated mean serial interval

Value

A `ggplot2` object that can be further customized or displayed. The plot includes appropriate axis labels, legend, and styling for publication-quality figures

References

Vink MA, Bootsma MCJ, Wallinga J (2014). Serial intervals of respiratory infectious diseases: A systematic review and analysis. American Journal of Epidemiology, 180(9), 865-875.

See Also

`si_estim` for serial interval estimation, `f_norm` and `f_gam` for the underlying mixture distribution functions

Examples

```
# Example 1: Visualize fit for simulated outbreak data
set.seed(123)
# Simulate ICC intervals from mixed distribution
icc_data <- c(
  rnorm(20, mean = 0, sd = 2),      # Co-primary cases
  rnorm(50, mean = 12, sd = 3),     # Primary-secondary cases
  rnorm(20, mean = 24, sd = 4)      # Primary-tertiary cases
)
icc_data <- round(pmax(icc_data, 0))  # Ensure non-negative

# Plot with estimated parameters
plot_si_fit(
  dat = icc_data,
  mean = 12.5,
  sd = 3.2,
  weights = c(0.2, 0.6, 0.15, 0.05),
  dist = "normal"
)
```

```
# Example 2: Using gamma distribution
plot_si_fit(
  dat = icc_data,
  mean = 12.0,
  sd = 3.5,
  weights = c(0.25, 0.65, 0.10),
  dist = "gamma",
  scaling_factor = 0.8
)
```

`si_estim` *Estimate Serial Interval Distribution Using the Vink Method*

Description

Estimates the mean and standard deviation of the serial interval distribution from outbreak data using the Expectation-Maximization (EM) algorithm developed by Vink et al. (2014). The serial interval is defined as the time between symptom onset in a primary case and symptom onset in a secondary case infected by that primary case.

Usage

```
si_estim(dat, n = 50, dist = "normal", init = NULL)
```

Arguments

`dat`	numeric vector; index case-to-case (ICC) intervals in days. These are calculated as the time difference between symptom onset in each case and symptom onset in the index case (case with earliest onset). Must contain at least 2 values. Values should be non-negative in most epidemiological contexts, though negative values are allowed for normal distribution
`n`	integer; number of EM algorithm iterations to perform. More iterations generally improve convergence but increase computation time. Defaults to 50, which is typically sufficient for convergence
`dist`	character; the assumed parametric family for the serial interval distribution. Must be either: • `"normal"` (default): Allows negative serial intervals, uses 7 mixture components • `"gamma"`: Restricts to positive serial intervals, uses 4 mixture components
`init`	numeric vector of length 2; initial values for the mean and standard deviation to start the EM algorithm. If `NULL` (default), uses the sample mean and sample standard deviation of the input data. Providing good initial values can improve convergence, especially for challenging datasets

Details

The Vink method addresses the challenge that individual transmission pairs are typically unknown in outbreak investigations. Instead, it uses index case-to-case (ICC) intervals - the time differences between the case with earliest symptom onset (index case) and all other cases - to infer the underlying serial interval distribution through a mixture modeling approach.

Methodological Approach:

The method models ICC intervals as arising from a mixture of four transmission routes:

- **Co-Primary (CP)**: Cases infected simultaneously from the same source
- **Primary-Secondary (PS)**: Direct transmission from index case
- **Primary-Tertiary (PT)**: Second-generation transmission
- **Primary-Quaternary (PQ)**: Third-generation transmission

The EM algorithm iteratively:

1. **E-step**: Calculates the probability that each ICC interval belongs to each transmission route component
2. **M-step**: Updates the serial interval parameters (mean, standard deviation) and component weights based on these probabilities

Distribution Choice:

- **Normal distribution**: Allows negative serial intervals (useful for modeling co-primary infections) and uses 7 components (positive and negative pairs for PS, PT, PQ routes plus CP)
- **Gamma distribution**: Restricts to positive values only, uses 4 components (CP, PS, PT, PQ without negative pairs). Recommended when negative serial intervals are epidemiologically implausible

Key Assumptions:

- The case with earliest symptom onset is the index case
- Transmission occurs through at most 4 generations
- Serial intervals follow the specified parametric distribution
- Cases represent a single, homogeneously-mixing outbreak

Input Data Preparation:

To prepare ICC intervals from outbreak data:

1. Identify the case with the earliest symptom onset date (index case)
2. Calculate the time difference (in days) between each case's onset date and the index case onset date
3. The resulting values are the ICC intervals for input to this function

Convergence and Diagnostics:

The EM algorithm typically converges within 20-50 iterations. Users should:

- Examine the fitted distribution using `plot_si_fit`
- Consider alternative distribution choices if fit is poor
- Try different initial values if results seem unreasonable
- Ensure adequate sample size (generally >20 cases recommended)

Value

A named list containing:

- `mean`: Estimated mean of the serial interval distribution (days)
- `sd`: Estimated standard deviation of the serial interval distribution (days)
- `wts`: Numeric vector of estimated component weights representing the probability that cases belong to each transmission route. Length depends on distribution choice (7 for normal, 4 for gamma)

References

Vink MA, Bootsma MCJ, Wallinga J (2014). Serial intervals of respiratory infectious diseases: A systematic review and analysis. American Journal of Epidemiology, 180(9), 865-875. doi:10.1093/aje/kwu209

See Also

`plot_si_fit` for diagnostic visualization, `integrate_component` for the underlying likelihood calculations

Examples

```
# Example 1:Basic usage with simulated data
set.seed(123)
simulated_icc <- c(
  rep(1, 20),   # Short intervals (co-primary cases)
  rep(2, 25),   # Medium intervals (primary-secondary)
  rep(3, 15),   # Longer intervals (higher generation)
  rep(4, 8)
)

result <- si_estim(simulated_icc)


# Example 2: Larger simulated outbreak, specifying distribution
large_icc <- c(
  rep(1, 38),   # Short intervals (co-primary cases)
  rep(2, 39),   #
  rep(3, 30),   # Medium intervals (primary-secondary)
  rep(4, 17),   #
  rep(5, 7),    # Longer intervals (higher generation)
  rep(6, 4),
  rep(7, 2)
)

result_normal <- si_estim(large_icc, dist = "normal")
result_gamma <- si_estim(large_icc, dist = "gamma")

# Example 3: Using custom initial values
result_custom <- si_estim(large_icc, dist = "normal", init = c(3.0, 1.5))
```

```
# Example 4: Specify iterations
result_iter <- si_estim(large_icc, n=100)
```

`smooth_estimates` *Apply Moving Average Smoothing to R Estimates*

Description

Applies temporal smoothing to reproduction number estimates using a centered moving average window. Handles missing and infinite values appropriately.

Usage

```
smooth_estimates(r_estimate, window)
```

Arguments

`r_estimate`	numeric vector; reproduction number estimates to smooth. Can contain NA or infinite values
`window`	integer; size of the smoothing window in time units. Window is centered around each point

Value

numeric vector; smoothed reproduction number estimates of the same length as input. Returns NA for points with insufficient valid neighboring values

---

`wallinga_lipsitch` *Estimate Time-Varying Case Reproduction Number Using Wallinga-Lipsitch Method*

Description

Estimates the time-varying case reproduction number (R_c) from daily incidence data using the method developed by Wallinga and Lipsitch (2007). The case reproduction number represents the average number of secondary infections generated by cases with symptom onset at time t, making it useful for retrospective outbreak analysis.

Usage

```
wallinga_lipsitch(
  incidence,
  dates,
  si_mean,
  si_sd,
  si_dist = "gamma",
  smoothing = 0,
  bootstrap = FALSE,
  n_bootstrap = 1000,
  conf_level = 0.95,
  shift = FALSE
)
```

Arguments

incidence	numeric vector; daily case counts. Must be non-negative integers or counts. Length must match dates
dates	vector; dates corresponding to each incidence count. Must be the same length as incidence. Can be Date objects or anything coercible to dates
si_mean	numeric; mean of the serial interval distribution in days. Must be positive. Typically estimated from contact tracing data or literature
si_sd	numeric; standard deviation of the serial interval distribution in days. Must be positive
si_dist	character; distribution family for the serial interval. Options: • "gamma" (default): Recommended for most applications as it naturally restricts to positive values • "normal": Allows negative serial intervals, useful when co-primary infections are suspected
smoothing	integer; window size for temporal smoothing of R estimates. Use 0 for no smoothing (default), or positive integers for moving average smoothing over the specified number of days
bootstrap	logical; whether to calculate bootstrap confidence intervals. Defaults to FALSE. Setting to TRUE increases computation time but provides uncertainty quantification
n_bootstrap	integer; number of bootstrap samples when bootstrap = TRUE. More samples provide more stable intervals but increase computation time. Defaults to 1000
conf_level	numeric; confidence level for bootstrap intervals, between 0 and 1. Defaults to 0.95 (95% confidence intervals)
shift	logical; whether to shift R estimates forward by one mean serial interval. When TRUE, adds a shifted_date column for comparison with instantaneous reproduction number estimates. Defaults to FALSE

Details

The method calculates the relative likelihood that each earlier case infected each later case based on their time differences and the serial interval distribution, then aggregates these likelihoods to estimate reproduction numbers. The approach makes minimal assumptions beyond specifying the serial interval distribution.

Key features:

- **Pairwise likelihood approach**: Considers all epidemiologically plausible transmission pairs (earlier to later cases)
- **Right-truncation correction**: Adjusts for unobserved future cases (see `calculate_truncation_correction`)
- **Bootstrap confidence intervals**: Quantifies estimation uncertainty
- **Temporal shifting**: Optional alignment with instantaneous R estimates
- **Flexible smoothing**: User-controlled temporal smoothing of estimates

The Wallinga-Lipsitch method estimates the case reproduction number by:

1. Computing transmission likelihoods from each earlier case to each later case based on the serial interval distribution
2. Normalizing these likelihoods so they sum to 1 for each potential infectee
3. Aggregating normalized likelihoods to estimate expected secondary cases per primary case
4. Applying corrections for right-truncation bias

Right-truncation correction accounts for secondary cases that may occur after the observation period ends (see `calculate_truncation_correction`). This correction is particularly important for recent cases in the time series.

Bootstrap confidence intervals are calculated by resampling individual cases with replacement, providing non-parametric uncertainty estimates that account for both Poisson sampling variation and method uncertainty.

Value

A data frame with the following columns:

- `date`: Original input dates
- `incidence`: Original input case counts
- `R`: Estimated case reproduction number
- `R_corrected`: Case reproduction number with right-truncation correction
- `R_lower`, `R_upper`: Bootstrap confidence intervals for R (if `bootstrap = TRUE`)
- `R_corrected_lower`, `R_corrected_upper`: Bootstrap confidence intervals for R_corrected (if `bootstrap = TRUE`)
- `shifted_date`: Dates shifted forward by mean serial interval (if `shift = TRUE`)

Note

The case reproduction number differs from the instantaneous reproduction number in timing: R_c reflects the reproductive potential of cases by their symptom onset date, while instantaneous R reflects transmission potential at the time of infection. Use `shift = TRUE` for comparisons with instantaneous R estimates.

References

Wallinga J, Lipsitch M (2007). How generation intervals shape the relationship between growth rates and reproductive numbers. Proceedings of the Royal Society B: Biological Sciences, 274(1609), 599-604. doi:10.1098/rspb.2006.3754

See Also

si_estim for serial interval estimation, generate_synthetic_epidemic for testing data, calculate_truncation_correct for right-truncation correction

Examples

```
# Example 1: Basic usage with synthetic data
set.seed(123)
dates <- seq(as.Date("2023-01-01"), by = "day", length.out = 30)
incidence <- c(1, 2, 4, 7, 12, 15, 18, 20, 22, 19,
               16, 14, 11, 9, 7, 5, 4, 3, 2, 1,
               rep(0, 10))

# Estimate reproduction number
result <- wallinga_lipsitch(
  incidence = incidence,
  dates = dates,
  si_mean = 7,
  si_sd = 3,
  si_dist = "gamma"
)

# View results
head(result)

# Example 2: With bootstrap confidence intervals
result_ci <- wallinga_lipsitch(
  incidence = incidence,
  dates = dates,
  si_mean = 7,
  si_sd = 3,
  si_dist = "gamma",
  bootstrap = TRUE,
  n_bootstrap = 500  # Reduced for faster example
)

# Plot results with confidence intervals
if (require(ggplot2)) {
  library(ggplot2)
  ggplot(result_ci, aes(x = date)) +
    geom_ribbon(aes(ymin = R_corrected_lower, ymax = R_corrected_upper),
                alpha = 0.3, fill = "blue") +
    geom_line(aes(y = R_corrected), color = "blue", size = 1) +
    geom_hline(yintercept = 1, linetype = "dashed", color = "red") +
    labs(x = "Date", y = "Reproduction Number",
         title = "Time-varying Reproduction Number") +
```

```
    theme_minimal()
}

# Example 3: With smoothing and shifting
result_smooth <- wallinga_lipsitch(
  incidence = incidence,
  dates = dates,
  si_mean = 7,
  si_sd = 3,
  si_dist = "gamma",
  smoothing = 7,  # 7-day smoothing window
  shift = TRUE    # Shift for comparison with instantaneous R
)

# Example 4: Using normal distribution for serial interval
result_normal <- wallinga_lipsitch(
  incidence = incidence,
  dates = dates,
  si_mean = 6,
  si_sd = 2,
  si_dist = "normal",
  smoothing = 5
)
```

`weighted_var` *Calculate Sample Weighted Variance*

Description

Computes the sample weighted variance of a numeric vector using precision weights. This function implements the standard unbiased weighted variance estimator commonly used in statistical applications where observations have different precisions or levels of confidence.

Usage

```
weighted_var(x, w, na.rm = FALSE)
```

Arguments

`x`	numeric vector; the data values for which to calculate weighted variance. Missing values are allowed if `na.rm = TRUE`
`w`	numeric vector; the precision weights corresponding to each observation in `x`. These represent how much confidence to place in each measurement (higher = more trusted). Must be the same length as `x`. Should be non-negative
`na.rm`	logical; if `TRUE`, missing values (both `NA` and `NaN`) are removed before computation. If `FALSE` (default), missing values will cause the function to return `NA`

Details

The weighted variance is calculated using the formula:

$$s_w^2 = \frac{\sum w_i}{\sum w_i^2 - (\sum w_i)^2} \sum w_i (x_i - \bar{x}_w)^2$$

where $\bar{x}_w$ is the weighted mean and w_i are the weights.

This function uses precision weights (also called reliability weights), which represent how much confidence or trust to place in each observation, rather than frequency weights that represent how many times to count each observation.

The denominator correction ($\sum w_i^2 - (\sum w_i)^2$) provides an unbiased estimator for precision weights. Examples of precision weights include probabilities (0-1), measurement confidence scores, or inverse error variances.

Value

numeric; the weighted sample variance. Returns `NA` if insufficient data or if `na.rm = FALSE` and missing values are present

See Also

`weighted.mean` for weighted mean calculation, `var` for unweighted sample variance

Examples

```
# Example 1: Basic weighted variance calculation
values <- c(2.1, 3.5, 1.8, 4.2, 2.9)
weights <- c(0.8, 0.3, 0.9, 0.5, 0.7)
weighted_var(values, weights)

# Example 2: Compare with unweighted variance
x <- 1:10
equal_weights <- rep(1, 10)
unweighted_var <- var(x)
weighted_var_equal <- weighted_var(x, equal_weights)

# Example 3: Using precision weights
# Measurements with different levels of confidence
measurements <- c(10.2, 9.8, 10.5, 9.9, 10.1)
confidence_levels <- c(0.9, 0.6, 0.8, 0.95, 0.7)  # How much we trust each measurement

precision_weighted_var <- weighted_var(measurements, confidence_levels)

# Example 4: Handling missing values
x_with_na <- c(1, 2, NA, 4, 5)
weights_with_na <- c(0.2, 0.3, 0.1, 0.8, 0.4)

# This will return NA
result_na <- weighted_var(x_with_na, weights_with_na, na.rm = FALSE)
```

```
# This will calculate after removing NA
result_removed <- weighted_var(x_with_na, weights_with_na, na.rm = TRUE)

# Example 5: Different weight patterns and their effects
data_points <- c(10, 15, 20, 25, 30)

# Equal weights (should approximate unweighted variance)
equal_wts <- rep(1, 5)
var_equal <- weighted_var(data_points, equal_wts)

# Emphasizing central values
central_wts <- c(0.1, 0.3, 1.0, 0.3, 0.1)
var_central <- weighted_var(data_points, central_wts)

# Emphasizing extreme values
extreme_wts <- c(1.0, 0.1, 0.1, 0.1, 1.0)
var_extreme <- weighted_var(data_points, extreme_wts)
```

`wt_loglik` *Calculate Weighted Negative Log-Likelihood for Gamma Distribution Parameters*

Description

Computes the weighted negative log-likelihood for gamma distribution parameters in the M-step of the EM algorithm for serial interval estimation. This function is used as the objective function for numerical optimization when the serial interval distribution is assumed to follow a gamma distribution.

Usage

```
wt_loglik(par, dat, tau2)
```

Arguments

`par`	numeric vector of length 2; parameters to optimize where `par[1]` is the mean and `par[2]` is the standard deviation of the serial interval distribution
`dat`	numeric vector; index case-to-case (ICC) intervals. Zero values are replaced with 0.00001 to avoid gamma distribution issues at zero
`tau2`	numeric vector; posterior probabilities (weights) that each observation belongs to the primary-secondary transmission component. These are typically derived from the E-step of the EM algorithm

Details

The function converts mean and standard deviation parameters to gamma distribution shape and scale parameters, then calculates the weighted log-likelihood based on the posterior probabilities from the E-step. Returns the negative log-likelihood for minimization by `optim`.

This function is used internally by si_estim when dist = "gamma". The gamma distribution is parameterized using shape (k) and scale (theta) parameters derived from the mean and standard deviation:

- Shape: $k = \mu^2/\sigma^2$
- Scale: $\theta = \sigma^2/\mu$

The weighted log-likelihood is calculated as:

$$\sum_i \tau_{2,i} \log f(x_i|k,\theta)$$

where $f(x_i|k,\theta)$ is the gamma probability density function and $\tau_{2,i}$ are the weights from the E-step.

Value

numeric; negative log-likelihood value for minimization. Returns a large penalty value (1e10) if parameters result in invalid gamma distribution parameters (non-positive shape/scale) or non-finite likelihood values

Note

This function is primarily intended for internal use within the EM algorithm. Users typically won't call this function directly but rather use si_estim with dist = "gamma".

See Also

si_estim for the main serial interval estimation function, optim for the optimization routine that uses this function

Examples

```
# Example usage within optimization context
# Simulate some ICC interval data and weights
set.seed(123)
icc_intervals <- rgamma(50, shape = 2, scale = 3)  # True mean=6, sd=sqrt(18)
weights <- runif(50, 0.1, 1)
initial_params <- c(5, 4)
likelihood_value <- wt_loglik(initial_params, icc_intervals, weights)

# Example of parameter optimization

optimized <- optim(
  par = initial_params,
  fn = wt_loglik,
  dat = icc_intervals,
  tau2 = weights,
  method = "BFGS"
)

cat("Optimized mean:", optimized$par[1], "\n")
cat("Optimized sd:", optimized$par[2], "\n")
```

Index